Sami Kolari

Selainpohjainen 3D-kartta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

6.4.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Sami Kolari Selainpohjainen 3D-kartta 33 sivua 6.4.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	mediatekniikka
Ohjaaja	yliopettaja Harri Airaksinen

Insinöörityön tavoitteena oli tehdä selainpohjainen kolmiulotteinen kartta Metropolian Leppävaaran kampuksesta käyttäen WebGL-ohjelmistorajapintaa. Tehty sovellus oli suunniteltu toimimaan sekä käyttäjien omilla älypuhelimilla ja tietokoneilla että kiinteissä opastusnäytöissä koulun tiloissa. Kartasta pystyy hakemaan luokkahuoneita huoneen nimen tai luokkatunnuksen perusteella sekä selaamaan karttaa sulavasti multi-touch-eleiden tai hiiren avulla. Työn tilaajana oli Metropolia Ammattikorkeakoulu.

Sovelluksessa hyödynnettiin myös useita JavaScript-kirjastoja esimerkiksi hakutulosten täydennykseen ja kameran liikeratojen animointiin. Näiden käytöstä laadittiin esimerkkejä sekä omakohtaisia huomioita. Lisäksi opeteltiin 3D-mallinnusta rakennuksen mallintamisen näkökulmasta kuvantamista hyödyntäen. Käyttäjätestauksessa tuli ilmi käytettävyyteen liittyviä ongelmia, jotka karttasovelluksessa piti korjata.

Kauppakeskusten nykyisiä opastusnäyttöjä vertailtiin käyttäjäkokemuksen ja käytettyjen teknologioiden osalta. WebGL-kirjastojen eroavaisuuksia tutkittiin koodiesimerkein. Työssä pohdittiin selainpohjaisen grafiikan tulevaisuutta ja Flash-pohjaisten Digital Signage -ratkaisuiden vääjäämätöntä loppua.

Sovellus onnistui hyvin, ja se johti jatkokehitykseen Metropoliassa. Karttaan toivottiin lisäominaisuuksia, kuten luokkahuoneiden varaustilanteen reaaliaikainen tarkastelu.

Avainsanat	WebGL, 3D, HTML5, Three.js, kartta, Digital Signage, QR-koodi

Author Title Number of Pages Date	Sami Kolari 3D map in browser 33 pages 6 March 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Media Technology
Instructor	Harri Airaksinen, Principal Lecturer

The goal of this thesis was to make a 3D map that works in web browsers for Metropolia University of Applied Sciences using a WebGL-framework. The application was designed to work on users' own smart phones and computers but also in fixed information displays in school's premises. It was possible to search for classrooms by their names or class numbers from the map. The map also featured multi-touch gestures and mouse support for smooth navigation. The work was done for Metropolia University of Applied Sciences.

The application used several JavaScript-libraries for features such as autocompletion for search and to tween camera movement. Examples and insights of their usage were made. 3D-modelling was studied from the perspective of building modelling and photogrammetry. Also, user testing and its results are discussed at the end of the thesis.

The current information displays of shopping malls are compared by their user experience and technologies used. Study was made about the differences of WebGL-libraries and the future of web-based graphics. The work also dealt with the inevitable end of Flash-based Digital Signage solutions.

The application succeeded, and it lead to further development by Metropolia. More features such as real-time display of classrooms' usage level was hoped to be implemented to it.

Keywords	WebGL, 3D, HTML5, Three.js, map, Digital Signage, QR-code

Sisällys

1 Johdanto

Insinöörityön tarkoituksena on tehdä kolmiulotteinen karttasovellus, jonka avulla käyttäjän on helppo löytää haluttu tila rakennuksesta tilan tunnuksen tai nimen avulla. Sovellus koostuu interaktiivisesta selainpohjaisesta kartasta, jossa rakennuksen eri kerrokset ja osat on visualisoitu selkeästi. Kun sovellukseen syötetään halutun tilan nimi tai tunnus, sovellus näyttää käyttäjälle, missä päin rakennusta tila sijaitsee. Tavoitteena on tehdä sovelluksesta riippumaton päätelaitteesta, jolloin sitä voidaan käyttää sekä vierailijan tai uuden opiskelijan omassa puhelimessa että rakennuksen kiinteissä infonäytöissä. Kartta on kolmiulotteinen, jotta rakennuksen eri kerrosten ja siipien hahmottaminen olisi helppoa. Työn tilaajana on Metropolia Ammattikorkeakoulu, joka sopii työlle erityisen hyvin, sillä kampukset ovat suuria ja luokkahuoneita on paljon.

Yksi työn tärkeimpiä tavoitteita on tehdä sovelluksesta niin helppokäyttöinen ja selkeä, että käyttö onnistuu keneltä tahansa vaivattomasti ilman käyttökoulutusta. Valitusta rakennuksesta tulee myös tehdä kolmiulotteinen malli oikeilla mittasuhteilla. Mallin tulee olla mahdollisimman tarkka mutta myös niin kevyt, että se toimii kaikissa nykyaikaisissa tietokoneissa ja älypuhelimissa.

Raportissa tutkitaan aluksi kauppakeskuksissa käytössä olevia opastusnäyttöjä, jotka ovat toimineet inspiraationa tämän opinnäytetyön aiheelle. Siinä myös otetaan kantaa Metropolian nykyisiin opastuksiin ja siihen, miksi ne kaipaisivat uudistusta. Tämän jälkeen vertaillaan työn kannalta varteenotettavia WebGL-kirjastoja ja perustellaan lopullinen valinta. Raportissa käsitellään käytettävyyden ja käyttöliittymän suunnittelua, joka edelsi sovelluksen varsinaista koodausta.

Sovelluksen toimintalogiikka käydään läpi koodiesimerkein, ja työssä hyödynnetyt JavaScript-kirjastot esitellään. Kartan ja rakennuksen 3D-mallinnusta sekä työssä käytettyä kuvamittausta käsitellään pikaisesti. Lopussa myös kerrotaan, kuinka sovelluksen käyttäjätestauksessa esiintyi seikkoja, joita suunnitteluvaiheessa ei ollut otettu huomioon. Sovelluksen tulevaisuudesta ja jatkokehityksestä kerrotaan niiltä osin, kun raportin kirjoitushetkellä tiedetään.

2 Interaktiiviset kartat

2.1 Kauppakeskusten opastusnäytöt

Craneworks on Suomen johtava Digital Signageen keskittyvä yritys. Sen tekemiä karttasovelluksia on käytössä ostoskeskuksissa kuten Kampissa ja Citycenterissä sekä risteilyalus Silja Serenadella. [1.] Tutkimalla Kampin ostoskeskuksen verkkosivuilta löytyvää karttaa (kuva 1) verkkoinspektorilla selviää, että se on toteutettu WebGL-kirjasto Three.js:llä. Samaa karttaa hyödynnetään myös kauppakeskuksen kiinteissä infonäytöissä.

Kuva 1. Kampin ostoskeskuksen verkkosivuilta löytyvä Craneworksin tekemä kartta kauppakeskuksesta [34].

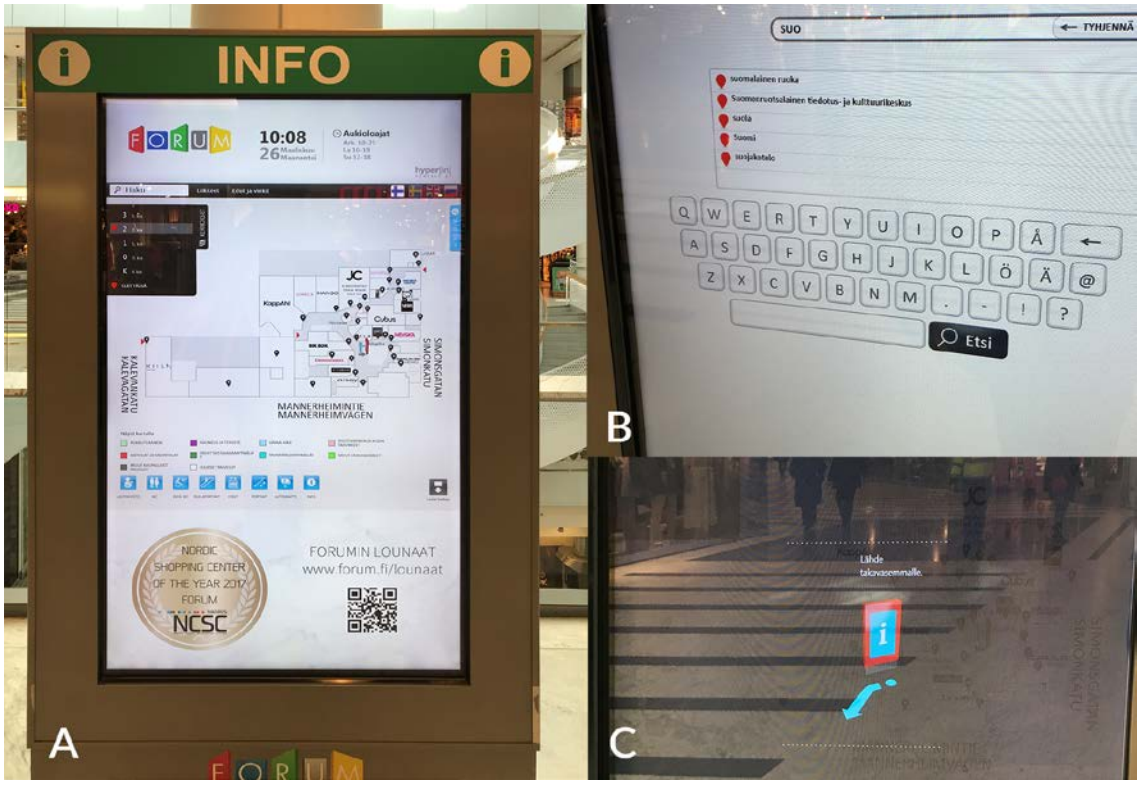
Kuvassa 1 näkyvä kartta pohjautuu Craneworksin BCN Dynamic Wayfinding -järjestelmään, jota yritys kuvailee seuraavasti:

> ”BCN Dynamic Wayfinding on keskitetty kauppapaikkojen ja julkisten tilojen info- ja opastusjärjestelmä, joka soveltuu muun muassa ostoskeskuksiin, messuhalleihin, julkisiin tiloihin, kampuksille ja laivoille. BCN wayfinding -palvelu opastaa asiakkaat etsimiensä tuotteiden ja palveluiden luokse. Järjestelmä näyttää sijaintisi kartalla, voit hakea tietoa eri tuotteista, kaupoista ja niiden tarjonnasta ympärilläsi. Kun sopiva tuote tai palvelu löytyy, järjestelmä ohjaa sinut lyhintä reittiä sen äärelle.” [1.]

Kuten kuvan 2 kohdassa A näkee, on Kampin kauppakeskuksessa näytöillä mainoksia, kunnes ruutua koskettaa. Kohdassa B on kartasta valittu kauppa aktiiviseksi sitä koskettamalla, jolloin näytön yläosaan ilmestyy tietoa kaupasta ja painike reittiohjeiden näyttämiseksi. Kohdassa C näkyy reitti piirrettynä lähtöpisteestä määränpäähän, ja jo määränpää on eri kerroksessa (kohta D), piirtyy viiva lähimmälle hissille. Hissin kuvaketta painamalla näkymä hyppää määränpään kerrokseen ja reitti piirtyy siellä loppuun asti. Sovelluksen kartta on käännetty samoin päin infonäytön ja täten myös käyttäjän kanssa, mikä auttaa käyttäjää hahmottamaan, mihin suuntaan hänen kuuluu lähteä. Sovelluksen hakutoiminto on kuitenkin hyvin epälooginen. Kuten kohdassa E näkyy, tulee hakusanalle ”suo” tuloksina ”Blue Lagoon” ja ”Dna Kauppa”. Kampissa on Suomalainen Kirjakauppa.

Kuva 2. Kampin infonäyttö ja sen eri ominaisuudet.

Kauppakeskus Forumilla on HyperIn Oy:n tekemä kartta. Yritys ei ole erikoistunut vain digital signage -ratkaisuihin, vaan se tarjoaa kattavasti palveluita kauppakeskusten asiakkaiden ja liikkeiden hallintaan ja analysointiin. [2.] Kuvassa 3 näkyvä kartta on perinteisempi kaksiulotteinen kerroskohtainen näkymä, jossa kauppojen omat logot on aseteltu liiketilojen päälle. Logoista löytää nopeasti hakemansa liikkeen, mutta tiheästi sijoitettujen pienten liiketilojen kohdalla muuttuu näkymä ahtaaksi ja hankalalukuiseksi. Karttaa ei myöskään ole käännetty käyttäjän suuntaiseksi toisin kuin Kampin tapauksessa, jolloin suuntien hahmottaminen voi olla työläämpää. Ratkaisuna tähän kartta kertoo suunnan, johon täytyy lähteä, kun hakee reittiohjeita (kohta C). Haku toimi hyvin.



Kuva 3. HyperInin tekemä kartta kauppakeskus Forumissa.

Kauppakeskus Forumin kartan kaltainen sovellus on käytössä myös kauppakeskus Kluuvissa. Kluuvin opasteet on tehnyt Upto Oy. [3.] Molemmissa karttapohjana on todennäköisesti kuvatiedosto, ja Kluuvin tapauksessa karttaa ei pysty suurentamaan. Karttoja ei voi kääntää, sillä tekstit näyttäisivät olevan kovakoodattuna karttakuvan päälle. Kumpikaan kosketusnäytöistä ei tukenut multi-touch -kosketusliikkeitä eikä sormen liu'uttamista näytön päällä esimerkiksi pitkien listojen selauksessa tai kartan liikuttelussa. Pitkien listojen selauksessa pitikin käyttää näytöllä olleita nuolinäppäimiä, joista painamalla lista liikkui alaspäin.

Craneworksin BCN Dynamic Wayfinding -järjestelmä oli käytössä Kampin lisäksi kauppakeskus Sellossa ja Citycenterissä, ja ne olivat toiminnaltaan hyvin samankaltaisia. Käyttöliittymäelementtien sijoittelua ja väripalettia on muutettu niin, että kartat sopivat kauppakeskusten graafisiin ilmeisiin. Kosketusnäytön toiminnallisuus on Craneworksin kartoissa kilpailijoita parempi, sillä sen toimintalogiikka multi-touch -kosketusliikkeineen on sama kuin älypuhelimien karttasovelluksissa. Esimerkiksi kartan suurentaminen kahdella sormella nipistämällä toimi yllättävän hyvin.

Jonatan Hildén kirjoittaa vuonna 2012 informaatiomuotoilua käsittelevässä blogissaan Kampin kauppakeskukseen perinteisten infotaulujen tilalle ilmestyneistä isoista kosketusnäytöistä [17]. Hän kertoo, että näytön reunassa on tarra, joka kehottaa käyttäjää koskettamaan ruutua aktivoidakseen infotaulun. Nykyään tällaista kehotetta ei löydy, koska kosketusnäytöt ovat jo hyvin vakiintunutta teknologiaa. Hän myös kritisoi näyttöjen hitaasti päivittyvää kuvaa ja matalaa tarkkuutta. Tilanne on kuitenkin muuttunut kuuden vuoden aikana huomattavasti, ja sekä infotaulut että niiden ohjelmistot ovat päivittyneet.

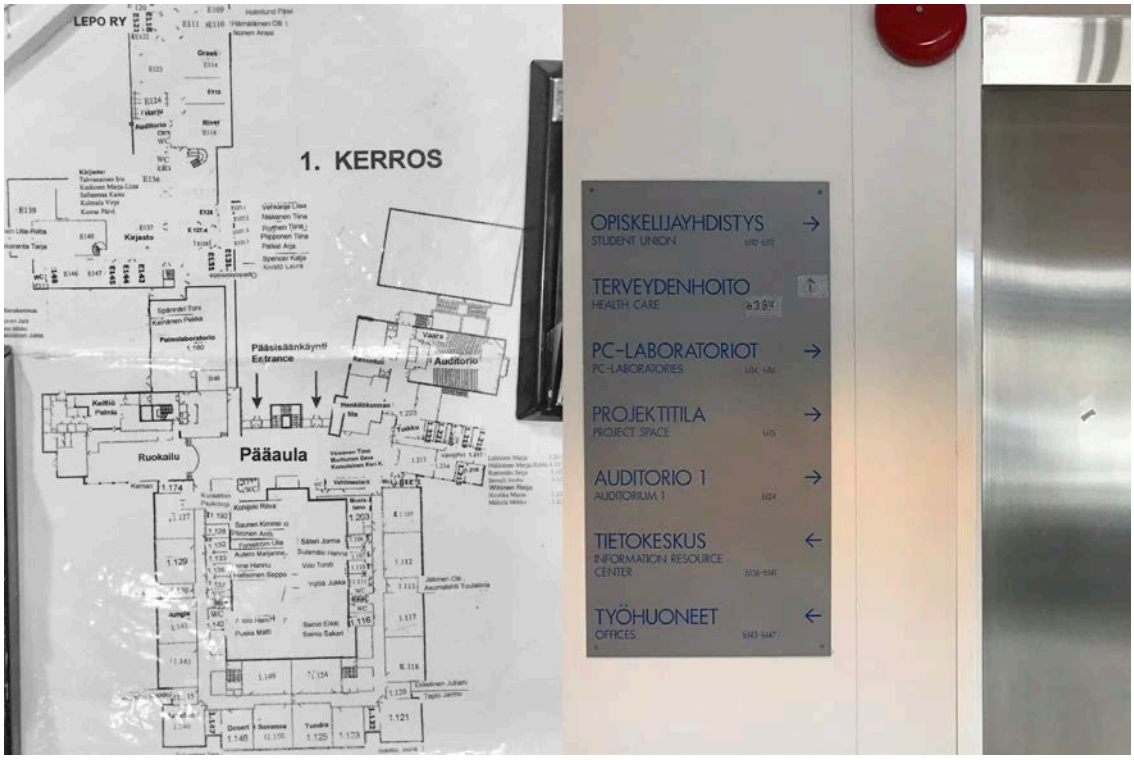
Kampin kauppakeskuksen nykyisten kosketusnäyttöjen valmistaja on latvialainen Saubag [1]. Yritys on erikoistunut Digital Signage -näyttöihin ja erityisesti ”toteemi”-tyyppisiin näyttötolppiin. Kosketusteknologiana voidaan käyttää joko infrapunaa tai kapasitiivisuutta. [18.] Infrapunakosketusnäyttö perustuu näytön reunaan sijoitettuun infrapunavaloon ja vastapäätä olevaan sensoriin, joka havaitsee säteen katkeamisen näiden välillä ja tulkitsee tämän kosketukseksi [19]. Kapasitiivinen kosketusnäyttö perustuu ihon kykyyn johtaa sähköä, jolloin kosketus aiheuttaa muutoksen näytön sähkökentässä [20]. Kampin tapauksessa uskoisin kyseessä olleen infrapunaan perustuva kosketusnäyttö, sillä näyttö rekisteröi kosketuksen jo ennen sormen osumista näyttöön.

Kolmas suosittu kosketusnäyttöteknologia on resistiivinen kosketuksentunnistus. Sen pinta koostuu kahdesta toisistaan hieman erillään olevasta kerroksesta. Kun käyttäjä koskettaa taipuisasta muovista tehtyä pintakerrosta, se taipuu ja muodostaa sähköisen kontaktin sisempään kerrokseen ja kosketus rekisteröityy. Etuna resistiivisissä kosketusnäytöissä on mahdollisuus käyttöön hanskojen kanssa, mikä ei kapasitiivisessa näytössä onnistu. Huonona puolena resistiivisessä kosketusnäytössä on usein huonompi tuntuma, sillä kosketuksen rekisteröitymiseksi täytyy käyttää hieman enemmän voimaa. Lisäksi näytön toiminnallisuus saattaa heikentyä tiheän käytön tai naarmuuntumisen seurauksena. [24.]

2.2 Leppävaaran kampuksen nykyiset opastukset

Metropolian Leppävaaran kampuksen rakennuksen seinillä on opasteita, kuten kuvan 4 oikealla puolella näkyy, jotka näyttävät lähellä olevat luokkahuoneet. Ongelmana tässä on se, että esimerkiksi koulun pääaulassa ei ole mitään opastusta, mistä päin koulun toisella puolella eri kerroksessa sijaitseva luokka ETYB313 löytyy. Vahtimestarilta kysymällä voi luokkahuonetta etsiä tulostetusta kartasta (kuvan 4 vasen puoli), mutta

koska rakennus on iso, on oikean huoneen löytäminen aikaa vievää. Kartta on myös huonolaatuinen ja joiltain osin lukukelvoton, eikä siitä vahtimestarin mukaan ole tallessa parempaa versiota.



Kuva 4. Vahtimestarilta löytyvä kartta sekä opaste ensimmäisessä kerroksessa.

Koska Metropolia opettaa tietotekniikkaa, olisi myös mielekästä, että sillä olisi esitellä näyttäviä ja teknisesti taidokkaita sovelluksia niin kampuksilla kuin koulun verkkosivuillakin. Karttasovellusta voisikin käyttää sekä koulun esittelysivulla vierailijoita että uusia opiskelijoita varten ja myös paikan päällä navigointia helpottamaan.

2.3 Toteutukseen soveltuvat teknologiat

Digital Signage -alalla on käynnissä suuri muutos Adoben lakkauttaessa Flash-alustansa tukemisen ja päivittämisen vuoden 2020 loppuun mennessä. Suurin vaikuttaja tähän on mobiililaitteiden huono tuki Flashille. Se on pakottanut kehittäjät miettimään vaihtoehtoisia ratkaisuja esimerkiksi verkkomainonnassa. [21.] Digital Signagessa on edelleen käytössä Flash-pohjaisia alustoja, sillä HTML5-pohjaiset alustat eivät ole vielä yhtä kehittyneitä ja vakiintuneita kuin 20 vuotta käytössä olleen Flashin. [22.]

HTML-canvaksessa pystytään piirtämään kaksi- ja kolmiulotteista grafiikkaa WebGL-rajapinnan avulla. Spesifikaation 1.0-versio julkaistiin vuonna 2011, eli kyseessä on melko tuore teknologia. [25.] WebGL-rajapintoja on useita, ja ne kaikki pohjautuvat Khronos Groupin ylläpitämän OpenGL:n kykyyn piirtää grafiikkaa näytölle. OpenGL on käyttöjärjestelmästä riippumaton avoin rajapinta näytönohjainkiihdytteisen grafiikan piirtämiseen. Juuri avoimuus ja laiteriippumattomuus mahdollistavat OpenGL:n standardoidun käytön kaikissa verkkoselaimissa. [26.] Muita grafiikan piirtoon tehtyjä rajapintoja on useita, kuten Microsoftin kehittämä DirectX sekä Vulkan, jonka kehityksestä vastaa myös Khronos Group. Vulkan on OpenGL:ää uudempi, tehokkaampi ja matalamman tason rajapinta. [27.] Vulkanille ja DirectX:lle ei ole kuitenkaan rakennettu tukea selaimiin OpenGL:n tapaan. DirectX on pelkästään Windows-ympäristöissä toimiva rajapinta, joten sen käyttäminen yleisenä grafiikkastandardina ei olisi mielekästä. [28.] ”WebVulkan” ei todennäköisesti tule myöskään olemaan WebGL:n seuraaja, sillä selaimessa toimiessaan rajapinta vaatii tiukasti määritellyt rajat tietoturvan takaamiseksi. WebGL ei esimerkiksi pääse käsiksi suoraan näytönohjaimen muistiin, ja Vulkanin suorituskykyetu perustuu osaltaan prosessorin ja näytönohjaimen väliseen suoraan kommunikointiin. [26.]

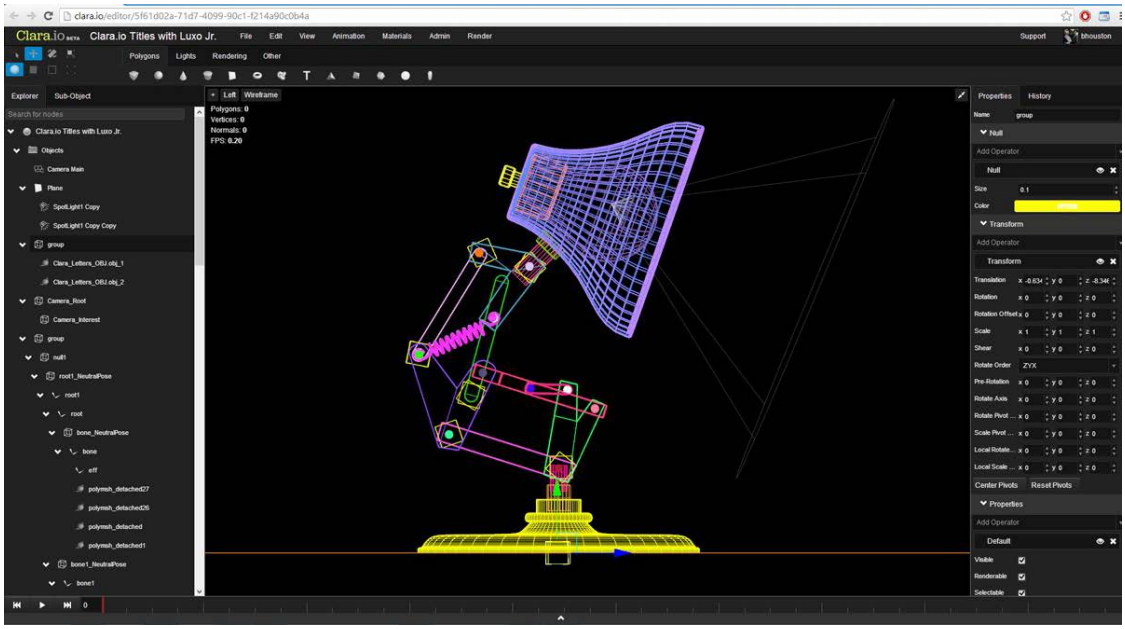
Tällä hetkellä kehitteillä oleva WebAssembly on herättänyt paljon huomiota, ja sen uskotaankin mullistavan selainpuolen web-kehityksen [29]. WebAssembly mahdollistaa matalan tason ohjelmointikielien kuten C:n kääntämisen suoraan selaimessa toimivaan muotoon. Näin saavutetaan huomattavasti parempi suorituskyky perinteiseen JavaScriptiin verrattuna, ja raskaat sovellutukset, kuten virtuaalikoneet ja koneoppiminen, ovat mahdollisia suoraan selaimessa. [30.] WebAssembly ei ole suoranaisesti grafiikan piirtoon suunniteltu teknologia, mutta mikään ei estä käyttämästä sitä siihen. Pelimoottori Unityn kehittäjät ovatkin toiveikkaita WebAssemblyn suhteen ja erityisen innoissaan siihen kehitteillä olevasta natiivista usean prosessoriytimen tuesta. [31.]

Päädyin työssäni WebGL-toteutukseen, sillä sovellus ei vaadi WebAssemblyn etuna olevaa raskasta laskentaa prosessorilta. Minulla ei myöskään ole kokemusta C-ohjelmoinnista, joten aikaa olisi kulunut huomattavasti enemmän. WebGL-rajapinnat toimivat pääpiirteittäin samankaltaisesti, ja suurimmat erot löytyvät lisätoiminnallisuuksista, palvelumallista sekä abstraktion tasosta. [4.] WebGL-kirjastoja ovat mm. Three.js (www.threejs.org), Playcanvas (www.playcanvas.com), Babylon.js (www.babylonjs.com) ja clara.io (www.clara.io). Lisäksi pelimoottori Unityllä tehtyjä projekteja voi kääntää WebGL-pohjaisiksi. [5.]

Three.js on vuonna 2009 alkunsa saanut avoimen lähdekoodin kirjasto, jonka tavoite on 3D-grafiikan ja animaation laaja-alainen tuki selaimessa. Kieli on abstraktiotasoltaan melko matalaa muihin WebGL-kirjastoihin nähden, eikä siinä ole esimerkiksi valmiita pohjia pelin tekemistä varten. Oletuksena mukana ei ole fysiikkamoottoria, mutta Three.js:lle on kehitetty fysiikkaliitännäinen Physijs, joka perustuu Ammo.js-kirjastoon, joka taas on suora JavaScript-käännös Bullet-fysiikkamoottorista. [32.] Kirjaston oppimisessa apuna toimivat lähinnä koodiesimerkit sekä dokumentaatio käytössä olevista konstruktoreista. Lisäksi kirjaston kehittäjät vastailevat aktiivisesti kysymyksiin Stack Overflow'ssa. Three.js on MIT-lisensoitu, eli sen kaupallinen käyttö on sallittua eikä omaa lähdekoodia tarvitse jakaa julkisesti. [4.]

Babylon.js on Microsoftin kehittämä kirjasto, joka julkaistiin ensimmäisen kerran vuonna 2013 Internet Explorer 11:n virallisena WebGL-rajapintana. Sen alkuperäinen käyttötarkoitus on pelinteossa, josta kertovat siihen natiivina kuuluvat törmäyksen tunnistus ja äänimoottori. Kirjaston oppimiseen on tehty kehittäjien toimesta ilmainen kahdeksantuntinen opetusvideokurssi. [6.]

Clara.io ja PlayCanvas ovat pidemmälle vietyjä palvelukokonaisuuksia, joihin kuuluvat yritysten omat julkaisualustat ja web-pohjaiset editorit (kuva 5). Molemmista on tarjolla ilmaisten versioiden lisäksi maksullisia versioita, joihin sisältyy lisäominaisuuksia, kuten yksityiset projektit, kasvatettu tallennustila ja puhelintuki. Projekteja voi kehittää ja säilyttää yritysten omilla sivustoilla, mutta ilmaisissa versioissa tehdyt projektit ovat aina julkisia. PlayCanvas sisältää myös sisäänrakennetun fysiikkamoottorin Ammo.js:n. [7; 8.]



Kuva 5. Clara.io sisältää kattavat selainpohjaiset 3D-mallinnustyökalut [7].

Kirjastojen yhtäläisyyksiä havainnollistamaan voidaan tarkastella esimerkkikoodia 1, jolla projektiin luodaan kuutio, määritellään siihen materiaali ja lisätään se lopuksi sceneen eli kolmiulotteiseen tilaan, johon projekti koostetaan.

```
Three.js:
var geometry = new THREE.CubeGeometry(10, 10, 10);
var material = new THREE.MeshLambertMaterial();
var cube = new THREE.Mesh(geometry, material);
scene.add(cube);


Babylon.js:
var cube = BABYLON.Mesh.CreateBox("cube", 2, scene);
var material = new BABYLON.StandardMaterial("material", scene);
cube.material = material;


PlayCanvas:
var cube = new pc.Entity();
cube.addComponent('model', {
     type: "box"
});
app.root.addChild(cube);
```

Esimerkkikoodi 1. Vertailu kuution luomisesta eri kirjastoilla.

Unity on suosittu pelimoottori, joka on vuodesta 2015 alkaen kehittänyt tukea projektien kääntämiselle WebGL-kielisiksi [5]. Aiemmin on Unity-projekteja voinut upottaa verkkosivuille käyttäen Unityn omaa selainlaajennusta. Selainlaajennuksen kehitys on kuitenkin lakkautettu, sillä Google, Microsoft ja Mozilla ovat parhaillaan ajamassa alas selaintensa NPAPI-laajennusten tukea, joihin Unityn laajennuskin kuului. [9.]

Toteutukseen valittiin Three.js-kirjasto, sillä se soveltuu mielestäni parhaiten insinöörityön projektiin keveytensä ansiosta ja sisältää kuitenkin kaikki tarvittavat ominaisuudet. Lisäksi minulla oli aikaisempaa kokemusta sen käytöstä. Kartan oli tarkoitus olla mahdollisimman kevyt, jotta se latautuu selaimessa yhtä nopeasti kuin mikä tahansa verkkosivu ja toimii sulavasti mahdollisimman lähellä laitteen natiivia ruudunpäivitysnopeutta. Three.min.js vie tilaa 513 kt siinä missä Unityn WebGL-käännös on huomattavasti raskaampi pelimoottorin laajuuden vuoksi. Three.js:n avoin lisenssi sopii myös projektiin hyvin, sillä karttaa on tarkoitus toimia suoraan Metropolian palvelimella.

3 Karttasovelluksen suunnittelu

3.1 Käyttöliittymä

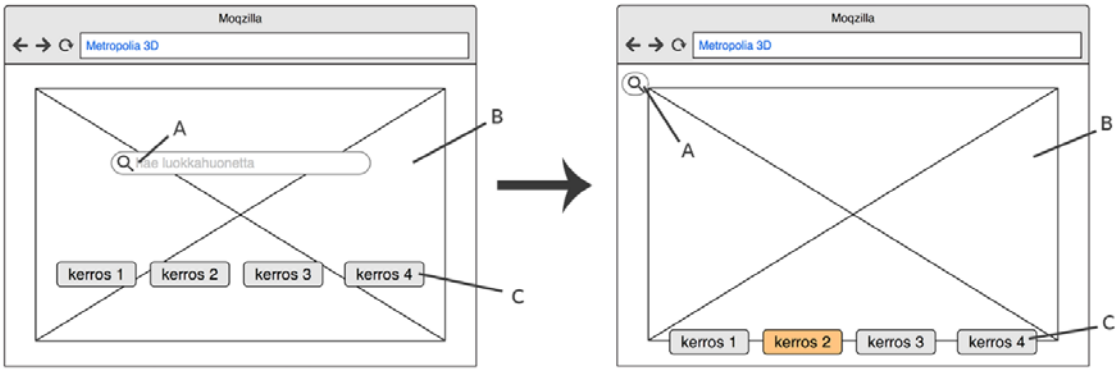
Steve Krug määrittelee käytettävyyden teoksessaan Don’t make me think seuraavasti:

> ” A person of average (or even below average) ability and experience can figure out how to use the thing to accomplish something without it being more trouble than it’s worth” [10, s. 9].

Käyttäjät eivät lue sivuston tekstejä vaan ”skannaavat” sivun pikaisesti katseellaan ja tekevät sen pohjalta johtopäätöksensä sivustosta [10, s. 22]. Tavoitteenani olikin, että karttasivustolle saapuessa on käyttäjälle välittömästi selvää, mikä sivun tarkoitus on ja miten sitä käytetään.

Sovelluksen käyttöliittymä suunniteltiin hyvin yksinkertaiseksi. Sivulle saavuttaessa hakupalkki (kuva 6, A) ja kerrosten valintanapit (C) ovat ainoat käyttöliittymäelementit näytöllä. Taustalle (B) sijoitetaan yleisnäkymä koulun julkisivusta. Se toimisi alkuruudun

taustakuvana ja muistuttaisi, mistä koulusta käyttäjä on itse asiassa luokkahuonetta hakemassa. Ensimmäisessä näkymässä sovellusta ei pitäisi olla mahdollista käyttää väärin, jos etusivulla ei ole muita painikkeita kuin hakupalkki ja painikkeet eri kerroksille.

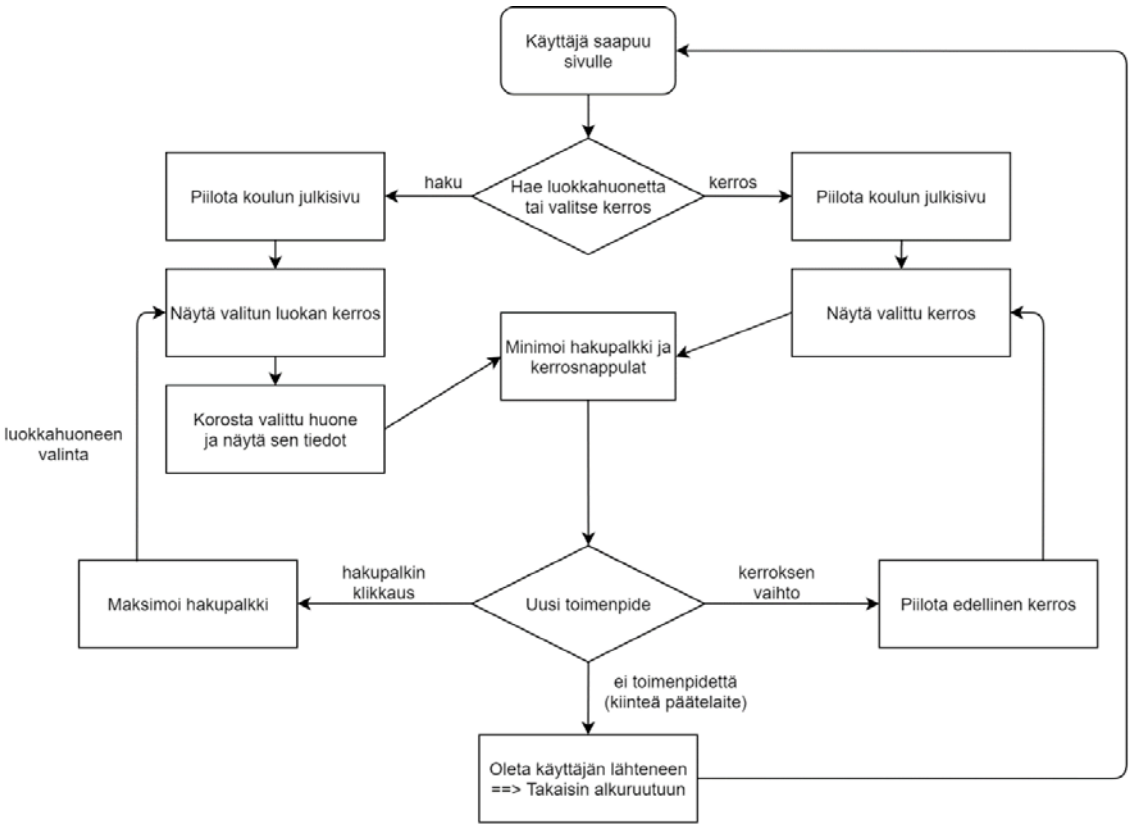


Kuva 6. Sivulle saavuttaessa hakupalkki (A) on keskiössä, sillä sivun tarkoitus on auttaa käyttäjää löytämään haluamansa luokkahuone. Vaihtoehtoinen käyttötapa on näyttää haluttu kerros ja selata sitä, ja tätä varten haun alapuolella on myös sille napit (C). Kun käyttäjä hakee huonetta tai valitsee kerroksen, jäävät haku ja kerroksen valinta toissijaisiksi toiminnoiksi. Huomio on tässä vaiheessa interaktiivisessa 3D-kartassa (B).

Kauhanen-Simanaisen mukaan verkkotilassa navigointi voidaan jakaa kahteen toimintatapaan: hakuun ja selailuun [11, s. 57]. Kerrospainikkeiden tarkoitus onkin mahdollistaa kartan vapaa tutkiminen, jos tavoitteena ei ole löytää mitään tiettyä luokkahuonetta vaan selata yleisesti koulun karttaa.

Kun käyttäjä on hakenut luokkahuonetta tai valinnut kerroksen, viedään haku- ja kerrosnapit häiriöttömämpiin sijainteihin ja antamaan lisää tilaa tässä vaiheessa toiminnan keskipisteeksi siirtyvälle kartalle. Kartassa liikutaan suoraan karttaa klikkailemalla tai koskettamalla. Jos käyttäjälle ei anneta muita vaihtoehtoja kartassa navigointiin (esimerkiksi klikattavia nuolinäppäimiä), alkaa käyttäjä arvioni mukaan klikkailla tai koskettaa karttaa ilman erillistä kehotusta.

Sovelluksen käyttölogiikan pitäisi olla selvää ilman ohjeiden lukemista. Sovelluksessa ei ole ”toimintoja toimintojen sisässä”, jolloin käyttäjä on aina yhden tai kahden klikkauksen päässä kaikista sovelluksen osa-aluista. Kuvassa 7 on kaavio sovelluksen suunnitellusta toimintalogiikasta.



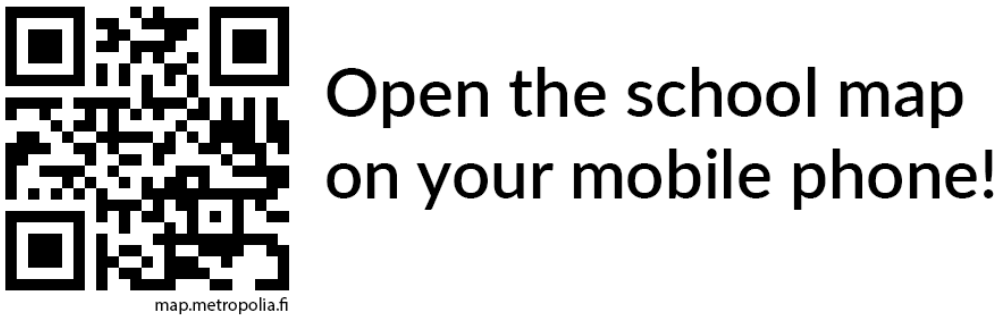
Kuva 7. Karttasovelluksen toimintakaavio.

3.2 Käyttäjän paikannus

GPS vaatii toimiakseen esteettömän näkyvyyden taivaalle, mikä ei koulun sisätiloissa ole mahdollista. GPS-signaali voidaan toistaa sisälle katolle asennettavan GPS-toistimen avulla, mutta silloinkin sijainniksi määrittyy vain toistimen sijainti eikä käyttäjän todellinen sijainti rakennuksen sisällä. [12.] Sisäpaikannus on mahdollista myös Bluetooth-lähettimien avulla, jolloin käyttäjän sijainti voidaan määrittää lähettimien ja käyttäjän etäisyyksien perusteella. Valitettavasti tämän mahdollistava rajapinta Web Bluetooth ei vielä ole standardoidussa käytössä verkkoselaimissa, joten paikannusta varten pitäisi karttasovelluksesta luoda natiivisovellus niin puhelimille kuin tietokoneillekin. [13.] Sama tilanne on WiFi Information API:n kanssa, joka on kehitteillä oleva selainrajapinta laitteen WiFi-signaalin hallintaan. Sen kautta signaalivahvuuksia mittaamalla lienee tulevaisuudessa mahdollista toteuttaa kolmiomittaukseen pohjautuva selaimessa toimiva sisätilapaikannus. [14.]

Karttasovelluksesta julkaistaan kaksi eri versiota. Ensimmäinen versio on tarkoitettu kiinteään päätelaitteeseen, joka voi olla esimerkiksi koulun aulaan asennettu kosketusnäyttö. Tässä tapauksessa sovellukseen on helppo kovakoodata ”olet tässä” -tyyppinen merkki haluttuun paikkaan kartalla. Jos aika riittää, tehdään myös mahdollisuus piirtää reitti valitun luokkahuoneen ja käyttäjän sijainnin välille.

Toinen versio on käyttäjän omalla laitteella toimiva verkkosovellus. Yksi mahdollisuus käyttäjän sijainnin määrittelemiseen kartalla olisivat tässä tapauksessa koulun seinille tulostetut QR-koodit, jotka avaavat karttasovelluksen lähettäen samalla parametrina luetun QR-koodin sijainnin. QR-koodien lukeminen on nopeampaa kuin osoitteen käsin kirjoittaminen, jos käyttäjällä vain on lukemista tukeva sovellus puhelimessaan. [15.] Näin ”olet tässä” -merkki voidaan sijoittaa huonekohtaisesti sovellukseen riippuen siitä, missä sijaitsevan QR-koodin käyttäjä lukee. Kuvassa 8 on esimerkki tällaisesta QR-koodista.



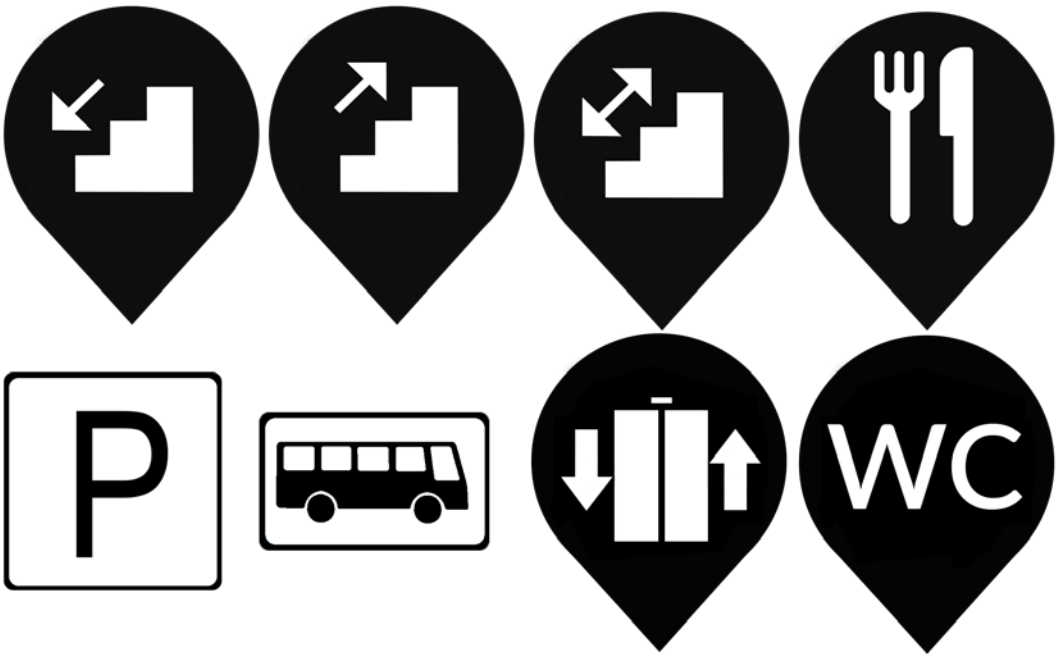
Kuva 8. QR-koodi vie osoitteeseen map.metropolia.fi/liikuntasali, jolloin sovellus voisi näyttää lähtöpisteen "olet tässä" -merkin liikuntasalissa.

3.3 Kartan graafinen ilme

Karttasovelluksen väriteema valittiin Metropolian graafisen ohjeiston värien mukaisesti [16]. Oranssin sävyt toimivat hyvin huomiovärinä valkoisia ja harmaita elementtejä vasten. Kuten Kampin kartan tapauksessa, kartan lattiaosuudet todennäköisesti määritetään huomiovärillä, tässä tapauksessa oranssilla, ja kartan huoneiden värit harmaalla. Näin huoneet ja lattia erottuvat toisistaan ja huonetta klikkaamalla se voidaan korostaa jollakin kirkkaalla värillä, esimerkiksi kirkkaan keltaisella. Kuvassa 9 ovat graafisen ohjeiston päävärit Pantone-, CMYK-, RGB- ja HTML/Hex-arvoina määritettyinä.

Kuva 9. Metropolian graafisen ohjeiston värit [16].

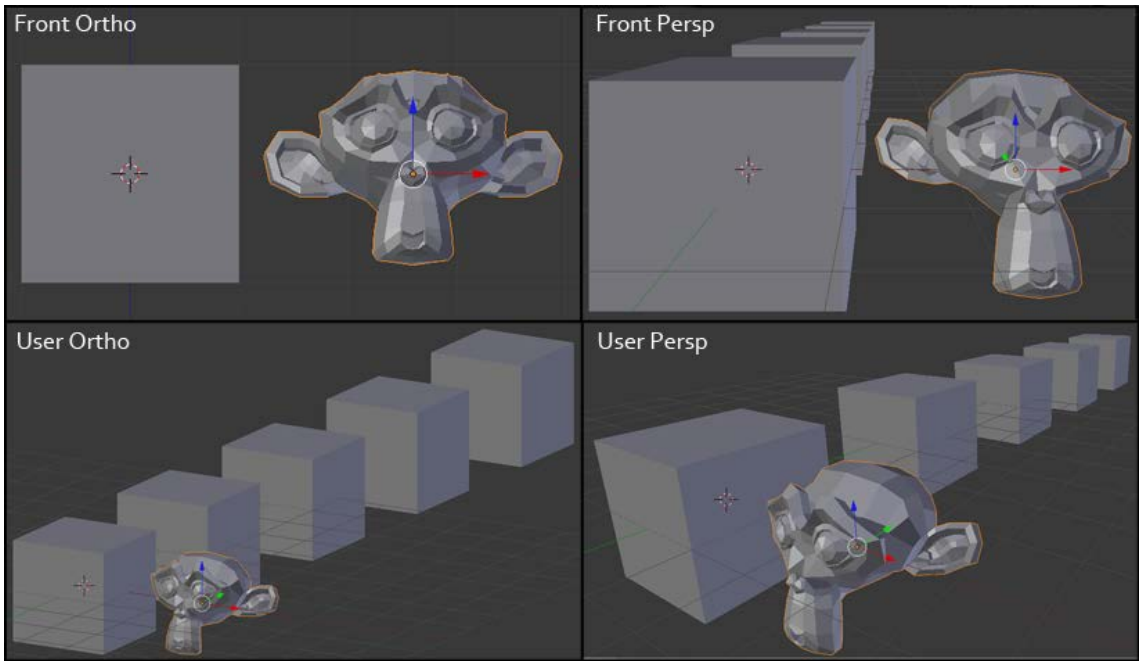
Kuvakkeita varten käytettiin mustaa ”nastaa” taustana. Nasta on leveä ja pyöreä, jotta sen sisään mahtuu kohdetta kuvastava kuvake mahdollisimman suurena luettavuuden säilyttämiseksi pienilläkin näytöillä. Kuvakkeet suunniteltiin esimerkiksi hissille, rappusille ja ruokalalle. Parkkipaikan ja bussipysäkin kuvakkeet eivät ole nastan sisällä, sillä niiden logot ovat hyvin vakiintuneet ja halusin säilyttää ne sellaisinaan. Lisäksi ne sijoitetaan koulun ulkopuolelle muiden ollessa sisäpuolella, jolloin sisä- ja ulkopuolen kuvakkeille muodostuu selkeä kahtiajako. Kuvassa 10 on tehtyjä kuvakkeita.



Kuva 10. Karttaa varten suunnitellut kuvakkeet.

Tässä vaiheessa on vaikea arvioida, kuinka tyyli tulee kokonaisvaltaisesti toimimaan. Värejä ja kuvakkeita on kuitenkin helppo testailla ja muuttaa projektin edetessä. Selaimen verkkoinspektorin kautta sovelluksen materiaalien muokkaaminen onnistuu ilman, että sovellusta tarvitsee edes ladata uudelleen. Näin väriyhdistelmien testailu on nopeaa ja alkuperäiseen tilanteeseen pystyy palaamaan yksinkertaisesti lataamalla sivun uudestaan.

Kartan visualisoinnissa pohdittava seikka on myös se, piirretäänkö näkymä perspektiivisesti vai ortografisesti. Perspektiiviä käytettäessä lähempänä olevat esineet piirtyvät suurempina kuin kauempana olevat. Näin saadaan helposti vaikutelma syvyydestä ja kolmiulotteisuudesta. Ortografisessa näkymässä kauempana olevat esineet näkyvät täsmälleen samankokoisina kuin lähempänä olevat samankokoiset esineet. Kuten värien kanssa, on myös perspektiivin vaikutusta helppo kokeilla itse sovelluksessa sen edetessä. Kuvassa 11 on havainnollisesttu selkeästi perspektiivisen ja ortografisen näkymän ero.



Kuva 11. Perspektiivisen ja ortografisen kameran ero [33].

3.4 Kosketuksen tunnistus

Hiiren lisäksi on kartassa pystyttävä navigoimaan kosketusnäytöllä. Tätä varten JavaScript tukee kosketustapahtumia, joiden avulla tiedetään, milloin käyttäjä aloittaa ja

lopettaa kosketuksen sekä missä sormi milläkin hetkellä on. Monta samanaikaista kosketusta hyödyntäviä eleitä tulkittaessa voi koodi paisua pitkäksi ja monimutkaiseksi, sillä esimerkiksi kahden sormen nipistys ja kahdella sormella pyöritys ovat kaksi yleisesti käytettyä elettä. Kahdella sormella pyörittäessä sormet saattavat tahattomasti liikkua lähemmäs toisiaan, jolloin ongelmaksi muodostuu tulkinta – oliko käyttäjän nipistysele tarkoituksenmukainen?

Eleiden hallintaa helpottamaan on JavaScript-kirjasto Hammer.js (https://hammerjs.github.io/), jolla eri eleiden tunnistus on helppoa. Sovellukseen voi sen avulla luoda toiminnallisuuksia, jotka aktivoituvat vaikkapa käyttäjän napauttaessa näyttöä kolmella sormella tai kaksoisnapauttaessa neljällä sormella. Karttasovelluksen kannalta keskeisiä eleitä ovat kahdella sormella pyöritys, josta Hammer.js:n avulla saadaan suoraan sormien välinen kulma, pyörityssuunta ja pyöritysnopeus. Kytkemällä arvot sovelluksen kameraan saadaan aikaan luonnollinen kartan kääntö kahdella sormella pyörittämällä. Myös kahden sormen nipistys tulee kytkeä kameran kykyyn siirtyä lähemmäksi tai kauemmaksi kartasta.

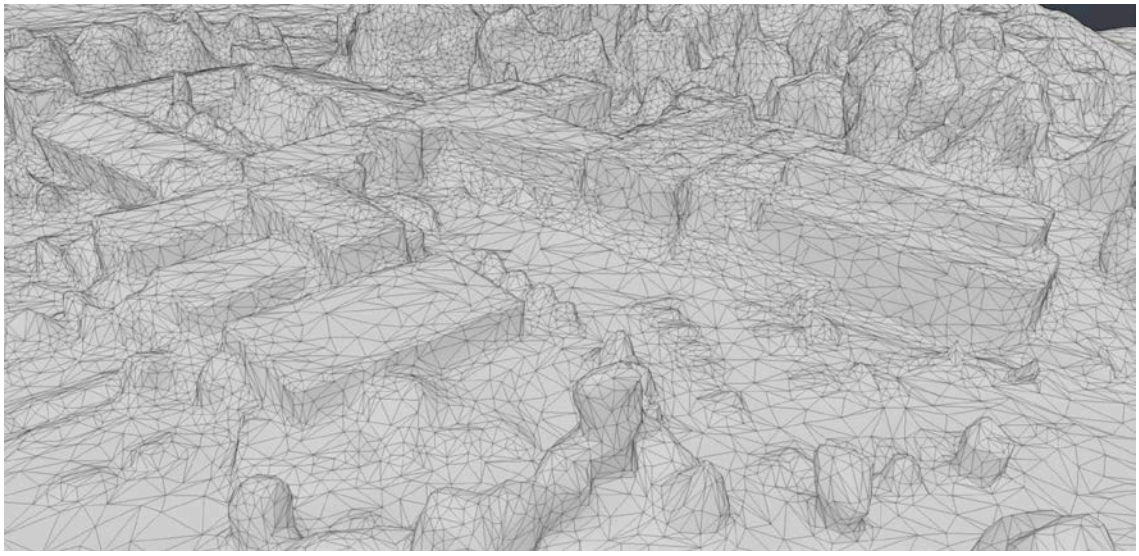
4 Karttasovelluksen toteutus

4.1 Koulurakennuksen mallinnus

Karttasovelluksen ensimmäistä testiversiota varten mallinnettiin yksinkertaiset 3D-mallit Metropolian Leppävaaran koulurakennuksen julkisivusta ja sen eri kerroksista. Mallin geometria on tehty mahdollisimman kevyeksi, jotta suorituskyky myös mobiililaitteilla pysyy hyvänä. Luokkahuoneet mallinnettiin käyttäen sopivankokoisia nelikulmioita, jotka nimettiin luokkatunnusten mukaan. Tällä tavoin sovelluksen hakutoiminto löytää halutun huoneen nimen perusteella kartalta.

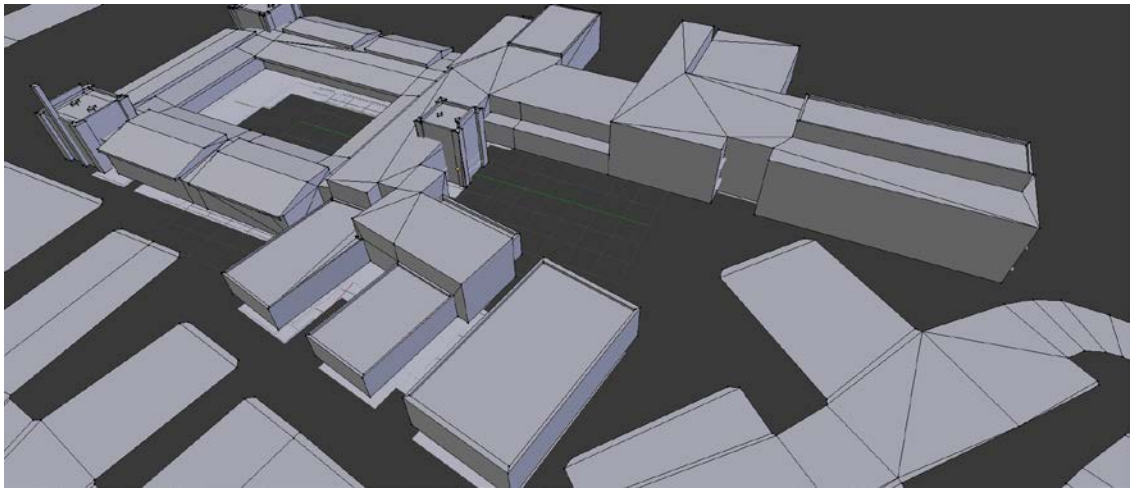
Mallinnusta helpottamaan tarvittiin rakennuksesta referenssimateriaalia. Perinteiset kartat eivät riittäneet, sillä koulun katot ovat eri korkeuksilla ja joidenkin osien katot ovat kaltevia. Helsingin kaupunki on toteuttanut laserkeilaskannatun tietomallin (www.hel.fi/3D), josta rakennusten 3D-malleja voi ladata vapaasti omaan käyttöön. Espoosta ei vastaavaa avointa mallia kuitenkaan ole. Google Mapsissa on tarkka malli myös Espoosta, mutta Googlen omistamia kaupunkimalleja ei pysty lataamaan. Ratkai-

suna toimi kuvantamisohjelma (photogrammetry) Autodesk ReCap, joka kokoaa 3D-mallin halutusta esineestä eri suunnilta otettujen valokuvien perusteella. Ohjelmaan syötettiin Google Mapsista otettuja kuvakaappauksia rakennuksesta eri kuvakulmista, ja näin saatiin aikaiseksi tarkka malli hyvillä mittasuhteilla. Saadun mallin geometria (kuva 12) ei ole kuitenkaan missään mielessä järkevä reaaliaikaiseen tarkasteluun, sillä se on tarpeettoman raskasta.



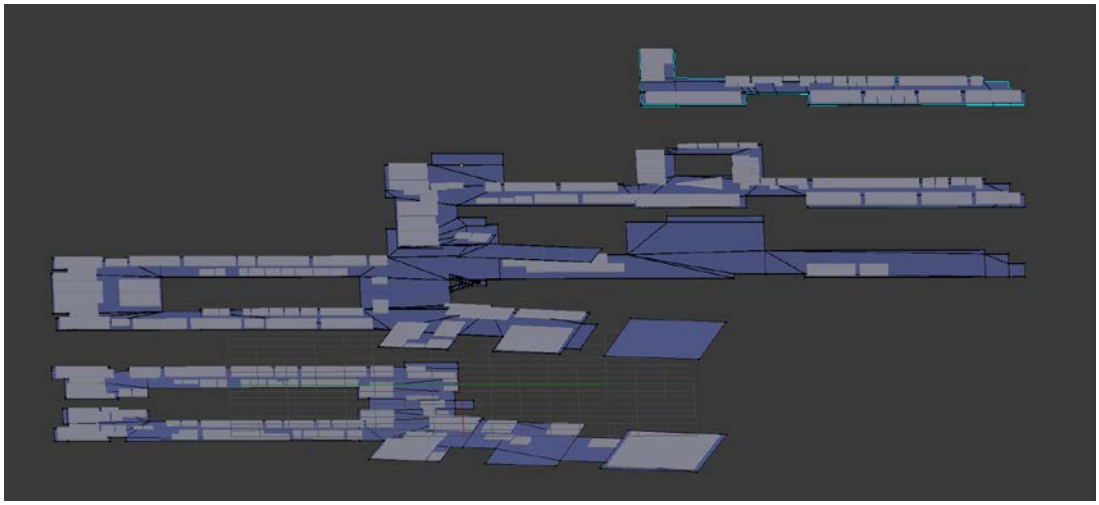
Kuva 12. Photogrammetriaohjelmalla tuotettu malli Metropolian Leppävaaran koulurakennuksesta.

Tämän jälkeen mallinnettiin käsin kuvantamismallia apuna käyttäen kevyt versio koulurakennuksesta. Näin mallin geometria keventyi noin 98 %. Tämänhetkinen tilanne on esiteltynä kuvassa 13.



Kuva 13. 3D-mallinnusohjelma Blenderissä mallinnettu koulurakennuksen julkisivu ja pysäköintipaikat.

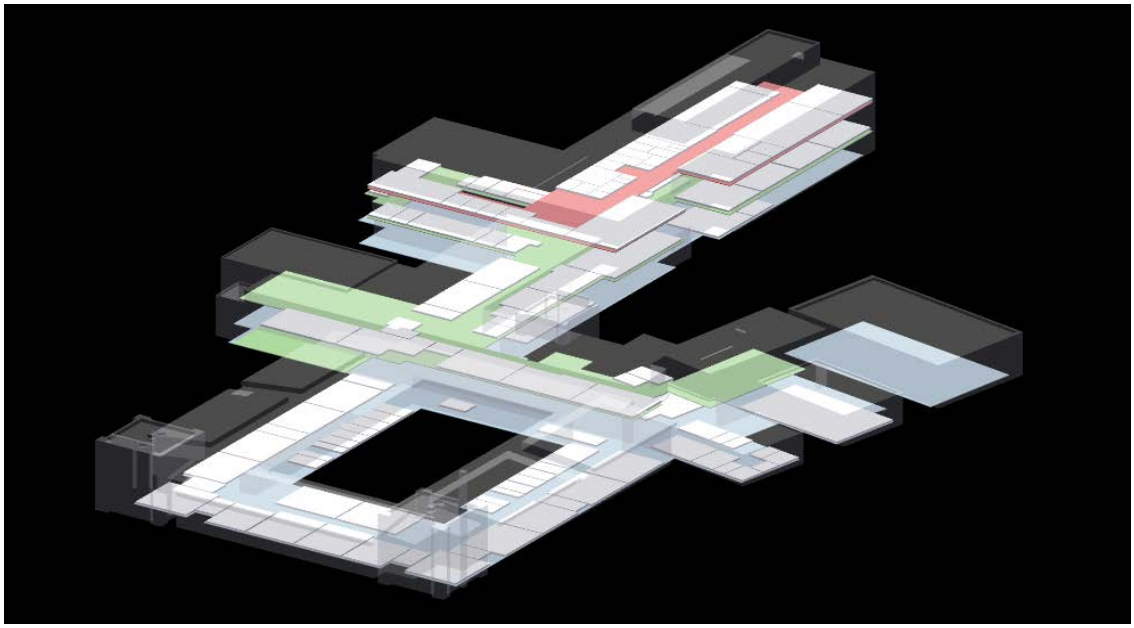
Koulurakennuksen kerrokset mallinnettiin vahtimestarilta saatujen kerroskuvien perusteella. Mittasuhteet olivat melko tarkat, mutta eivät kuitenkaan täydelliset: Kerrosmalleista (kuva 14) tuli hieman pidempiä ja kapeampia kuin julkisivumallin sisälle mahtui. Virhe voi johtua kuvantamisohjelman tuottamasta hieman vääristyneestä mallista tai vahtimestarin kerroskuvien vääristyneestä kuvasuhteesta. Vääristymä oli kuitenkin pieni, ja kartan on tarkoitus olla vain suuntaa antava, joten kerrosmallit sovitettiin sopimaan julkisivun sisälle litistämällä kerrosten pituutta n. 5 %. Luokkahuoneita kuvastavat objektit on nimetty luokan tunnuksen mukaan, jolloin haluttu huone voidaan Three.js:ssä valita aktiiviseksi luokkatunnuksella hakemalla.



Kuva 14. Koulurakennuksen neljä kerrosta ja luokkahuoneet mallinnettuina.

4.2 Toiminnallisuuden koodaus Three.js:llä

Seuraavaksi luotu 3D-malli tallennettiin WEBGL:n tukemaan .OBJ-muotoon ja se tuotiin näkyviin Three.js:llä luotuun projektiin. Näkymä oli tässä vaiheessa hyvin sekava, sillä lähekkäin toisiaan olevat kerrokset sekoittuivat keskenään ja karttaa oli vaikea tulkita. Kerrosten eteen piirtyvä talon ulkopinta oli myös hieman häiritsevä. Julkisivun materiaali reagoi valoon (Lambert shader), mutta kerrosten ja huoneiden materiaalina toimiva basic shader pysyi kuvakulmasta ja valon sijainnista huolimatta samanvärisenä. Näin kerros piirtyi selkeästi ja tasaisen värisenä. Kuvassa 15 näkyy tämänhetkinen tilanne.

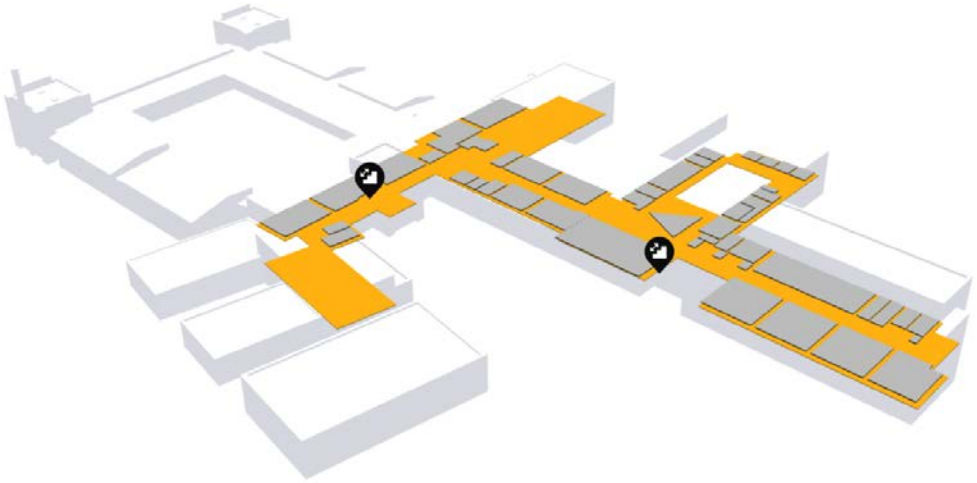


Kuva 15. Ensimmäinen koottu versio koulurakennuksesta selaimessa piirrettynä

Eri kerrosten läpinäkyvyyttä säädettiin niin, että vain valittu kerros näkyi täysin ja muut kerrokset olivat läpikuultavia. Tässä ongelmaksi muodostui OpenGL:n tapa käsitellä useita läpinäkyviä esineitä. Läpinäkyvän pikselin RGBA-arvo piirretään taustalla olevan pikselin RGB-arvon päälle, mutta tämä ei toimi, jos läpinäkyviä esineitä on useita päällekkäin. Jos läpinäkyviä esineitä laittaa useamman useita, piirtyy vain yksi lisätyistä läpinäkyvistä esineistä kiinteän esineen päälle. [23.]

Koska haluttiin säilyttää mahdollisuus piirtää kerroksia läpinäkyvinä, määritettiin sovellukseen kaksi erillistä piirtotasoa (render layers). Ensimmäisessä tasossa piirretään talon julkisivu ja seuraavassa halutut rakennuksen kerrokset ensimmäisen tason päälle. Mo-

lempiin tasoihin käytetään samaa kameraa, jolloin näkymät pysyvät keskenään kohdistettuina. Näin julkisivu näkyy aina vähemmän häiritsevästi kartan taustalla, eikä se rajoita läpinäkyvyyden käyttöä toisella piirtotasolla. Tämä toki tarkoittaa sitä, että rakennuksen kerroksia ei pysty piirtämään julkisivun taakse, vaan ne näkyvät aina sen edessä. Värit vaihdettiin myös Metropolian graafisen ohjeiston mukaisiksi. Edistys näkyy kuvassa 16. Tässä vaiheessa vaihdettiin myös kamera ortografisesta perspektiiviseksi, sillä koin kartan näyttävän näin mielenkiintoisemmalta ja selkeämmin hahmotettavalta.



Kuva 16. Koulurakennuksen kartta piirrettynä kahdessa tasossa Metropolian väreillä.

Jotta karttaan voi merkitä sijainteja ja kohteita, kuten portaita, lisättiin karttanäkymän päälle kuvakkeita perinteisinä HTML-elementteinä. Tällä tavoin kuvakkeiden koon määrittäminen on suoraviivaista CSS-tyyliä muokkaamalla ja ne varmasti piirtyvät sopivankokoisina erikokoisilla näytöillä. Tämä mahdollistaa myös muun HTML-sisällön upottamisen kartan päälle tehokkaasti, minkä avulla esimerkiksi päivän ruokalista voisi tulla näkyviin ruokalaa klikkaamalla. Halutut kuvakkeet syötetään sovellukseen listana, esimerkiksi ensimmäisen kerroksen kohdalla esimerkkikoodi 2:n tavoin:

```
floor1 = [
      ["icon_stairsboth", new THREE.Vector3(-.2,0,.6)],
      ["icon_here", new THREE.Vector3(-2.2,0,2.6)],
      ["icon_stairsup", new THREE.Vector3(-3.2,0,-10)],
      ["icon_stairsup", new THREE.Vector3(-3.7,0,-18)],
      ["icon_food", new THREE.Vector3(-9,0,.5)],
      ["icon_elevator", new THREE.Vector3(-6.2,0,-10)]
];
```

Esimerkkikoodi 2. Kuvakkeiden lisäys listana karttasovellukseen.

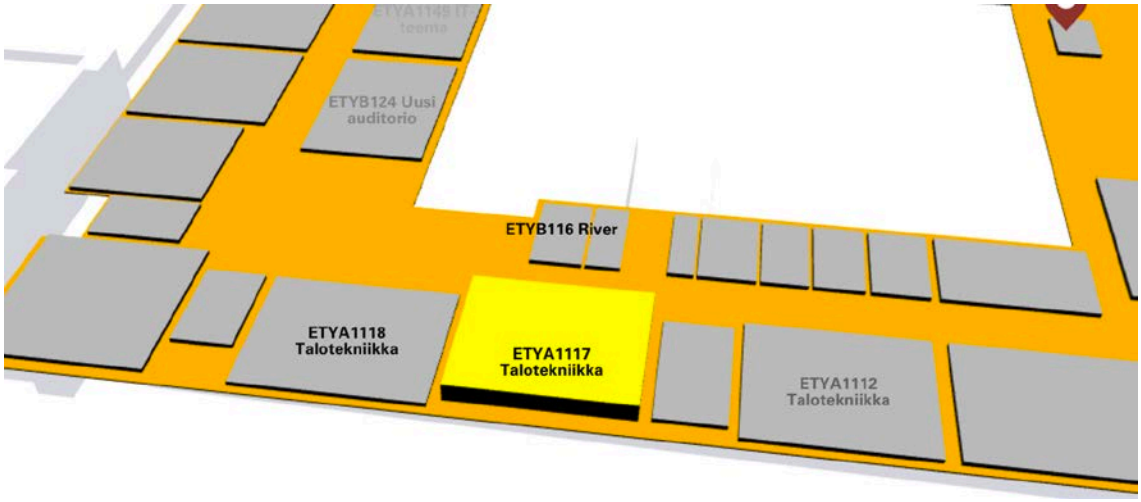
Kuvakkeen haluttu paikka syötetään XYZ-muotoisena vektorina ja sen sijainti näytöllä lasketaan jokaisella piirretyllä kuvalla Three.js:n metodilla vector.project(). Metodi palauttaa arvon muodossa X = (-1…1), Y = (-1…1), jolloin HTML-elementin sijainti voidaan päivittää muuttamalla elementin tyyliattribuutteja. Syöttämällä X- ja Y-arvot NDC-muotoon (normalized device coordinates eli normalisoitu koordinaatisto) muunnettuina CSS-attribuutteihin “top” ja “left” saadaan elementti liikkumaan halutulle paikalle. Tämä tapa on kuitenkin laskennallisesti hyvin raskas, ja usean elementin jatkuva liikuttaminen laski sovelluksen suorituskykyä. Käyttämällä CSS:n transform-arvoja päivitys on kuitenkin huomattavasti kevyempää. Jokaisen elementin ”todellinen” sijainti näytöllä on silloin staattinen (oletuksena top: 0 left: 0 eli ruudun vasen ylälaita), mutta elementin sisältöä liikutetaan pitäen elementin todellinen sijainti aina paikallaan. Koodi, jolla kuvakkeiden paikka päivitetään, on seuraavanlainen:

```
icons = eval("floor"+currentFloor+"Icons");
for (i = 0; i < icons .length; i++){
     var vector = icons[i][1].clone();
     vector.project(camera);
     vector.x = (vector.x + 1)/2 * window.innerWidth -25;
     vector.y = -(vector.y - 1)/2 * window.innerHeight +35;
     iconDivs[i].style.transform = "translate(" + vector.x + "px, " + vector.y
+ "px)";
}
```

Esimerkkikoodi 3. Kuvakkeiden sijainnin päivitykseen käytetty koodi.

Koodiesimerkissä määritetään elementin sijainti pikseleinä, vaikka se olisi myös mahdollista laskea prosentteina jättämällä arvot kertomatta window.innerWidthillä ja window.innerHeightilla. Pikseleissä on kuitenkin se etu, että ikonin nastan sijaintia voidaan säätää pikselitarkkuudella, jotta sen alaosan ”kärki” on merkityssä sijainnissa elementin vasemman ylälaidan sijaan.

Luokkahuoneita on niin paljon, että niiden nimet menevät päällekkäin, jos kaikki nimet ovat näkyvissä. Lisäksi kaikkien nimien näyttäminen ja niiden sijaintien päivittäminen on melko raskas operaatio. Ratkaisuna tähän päivitetään vain niiden luokkahuoneiden nimet, jotka ovat näkyvissä. Nimet näytetään ja sijainnit päivitetään, jos kamera on tarpeeksi alhaalla ja osoittaa suoraan luokkahuonetta kohti. Kuvan laidoilla olevat nimet häivytetään pois näkyvistä.

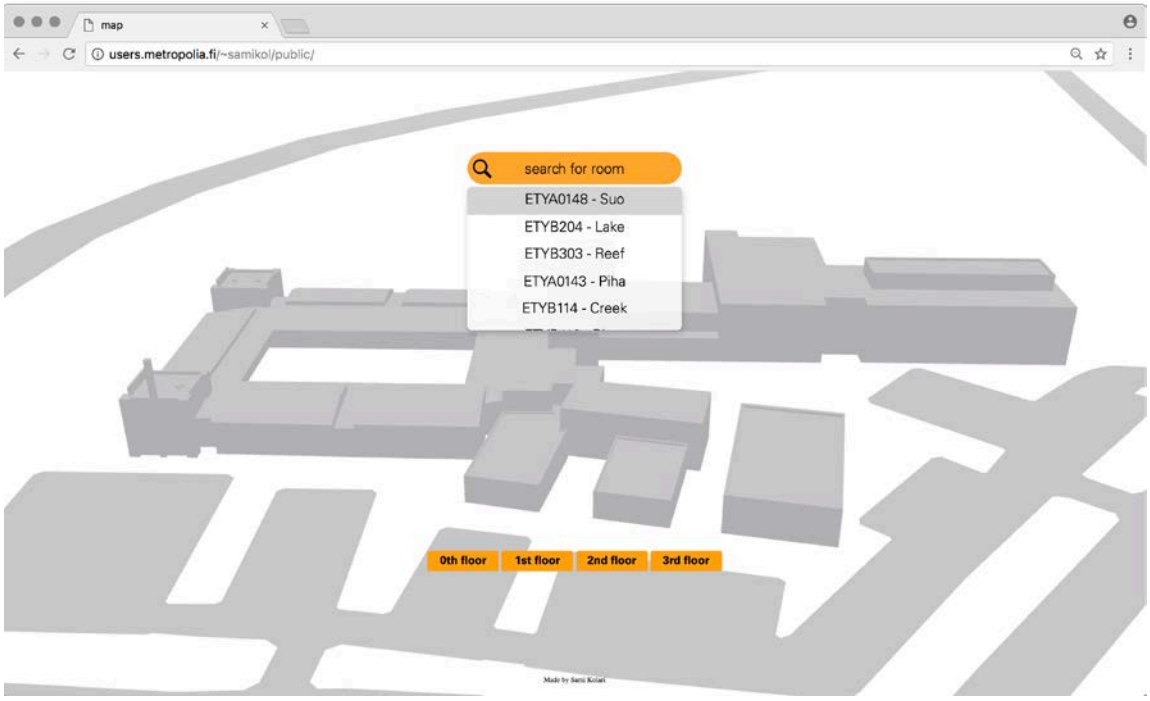


Kuva 17. Luokkahuoneiden nimet häivytetään, kun ne lähestyvät näytön reunaa.

4.3 Hakutoiminto ja muut painikkeet

Huoneenhakutoimintoon käytettiin Awesomplete.js-kirjastoa, joka on kevyt lisäosa tekstikentän automaattiseen täydennykseen. Sitä varten tehtiin luettelo kaikista Leppävaaran kampuksen luokkahuoneista osoitteesta wiki.metropolia.fi, ja ne syötetään listana hakupalkin alustuksessa.

Hakupalkin tyyli säädettiin teemaan sopivaksi oransseine väreineen. Kuvassa 18 näkyy luotu teema. Huone-ehdotusten listan korkeus määritettiin niin, että puhelimella käytettäessä näytölle ilmestyvä näppäimistö ei mene listan päälle. Awesompletessa on myös tapahtumia, joilla esimerkiksi huoneen vaihdon voi kytkeä käynnistymään ehdotuslistan elementin kosketuksella. Oletuksena listan elementin kosketus siirtää sen hakupalkkiin, minkä jälkeen käyttäjän täytyy vielä painaa rivinvaihtopainiketta. Oletusasetuksilla myöskään puhelimen näppäimistö ei haun jälkeen mennyt piiloon, mutta tämän pystyi korjaamaan Javascriptin blur-metodilla, jolla selaimen saa poistamaan kohdistuksen hakupalkista ja näin piilottamaan näppäimistön.



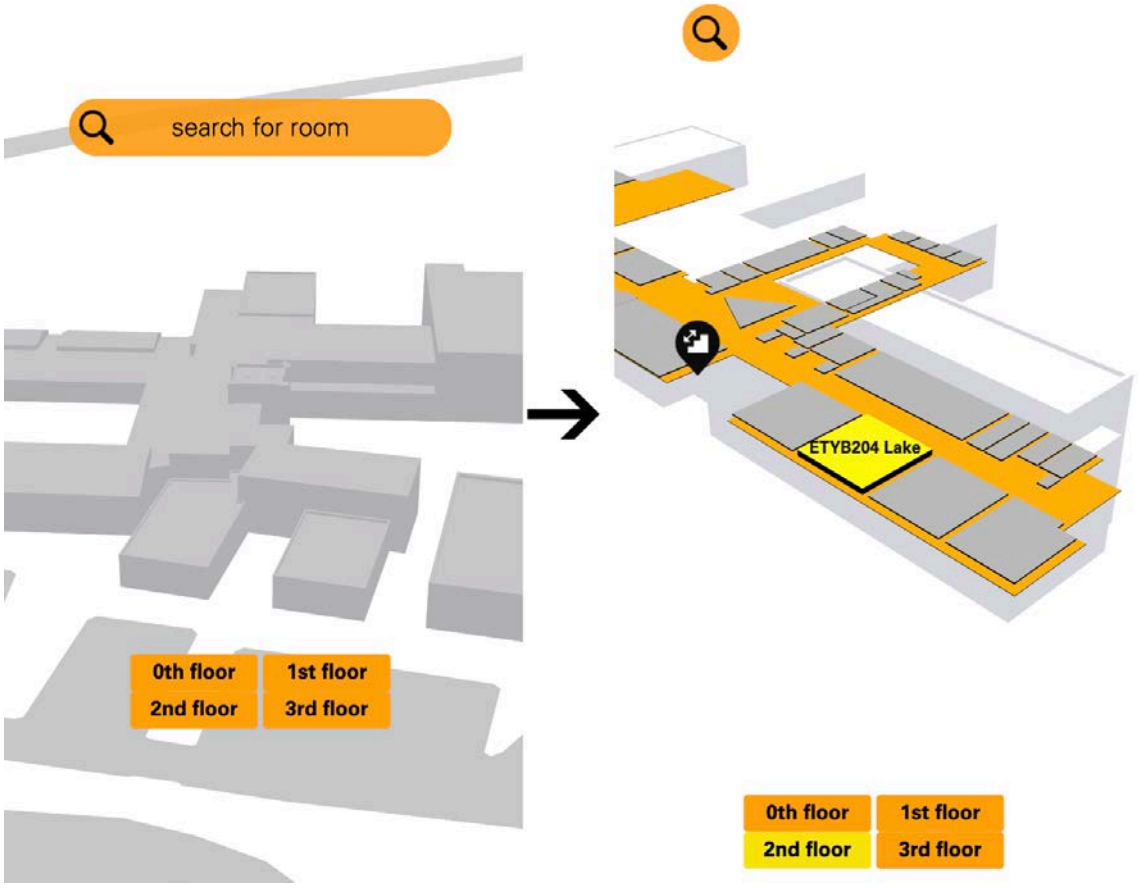
Kuva 18. Awesomplete.js-hakupalkki, jonka teema on mukautettu sovellukseen sopivaksi sekä kerrospainikkeet näkymän alalaidassa.

Kun käyttäjä valitsee huoneen, esimerkiksi ”ETYA204 – Lake”, parsitaan valinnasta kaikki muu paitsi numerot pois. Näin jäljelle jää ”204”, jonka sovellus osaa yhdistää samoin nimettyyn huoneeseen 3D-mallissa. Ensimmäinen numero kertoo oikean kerroksen. Sovellus vaihtaa ensin näkymän haluttuun kerrokseen, ja kun tämä on tehty, valitaan haluttu luokkahuone aktiiviseksi kyseisessä kerroksessa.

Haun jälkeen hakupalkki kutistuu palloksi näytön vasempaan laitaan (kuva 19), jolloin näytölle jää enemmän tilaa selaamiselle. Kutistumisen animoin jQueryn animate-metodilla, joka toimii CSS-atribuutteihin:

```
$("#awesomplete").animate({
     width: "-=255px",
}, 300, function() {
     searchMinimised = true;
});
```

Esimerkkikoodi 4. Hakupalkin kutistamisen animointi jQueryllä.



Kuva 19. Hakupalkin minimointi ja kerrospainikkeiden siirtäminen näytön alalaitaan karttatilan maksimoimiseksi.

4.4 Animoidut kamera-ajot

Rakennuksen kerrokset ovat erikokoisia ja -muotoisia, joten kunkin kerroksen kohdalla kameran sijainti piti määrittää käsin, jotta kerros täyttää kameran näkymän. Myös huonetta valittaessa pitää kameran liikkua kohti huonetta. Tähän tarvittiin sulavia kamera-ajoja sen sijaan, että kamera vain hyppäisi haluttuun sijaintiin. Tämä onnistuu Tween.js-kirjastolla, joka laskee kameran sijainnin eri ajanhetkillä alku- ja loppupisteiden välillä. Esimerkki sen käytöstä kameran liikutukseen on esimerkkikoodi 5:ssä.

```
var cameraEndPosition = {“x”: 10.5, “y”: 2, “z”: 5};
var cameraPositionTween = new TWEEN.Tween(camera.position).to(cameraEndPosi-
tion, 600). easing(TWEEN.Easing.Sinusoidal.InOut).onComplete(function(){tween-
Complete = true});
cameraPositionTween.start();
```

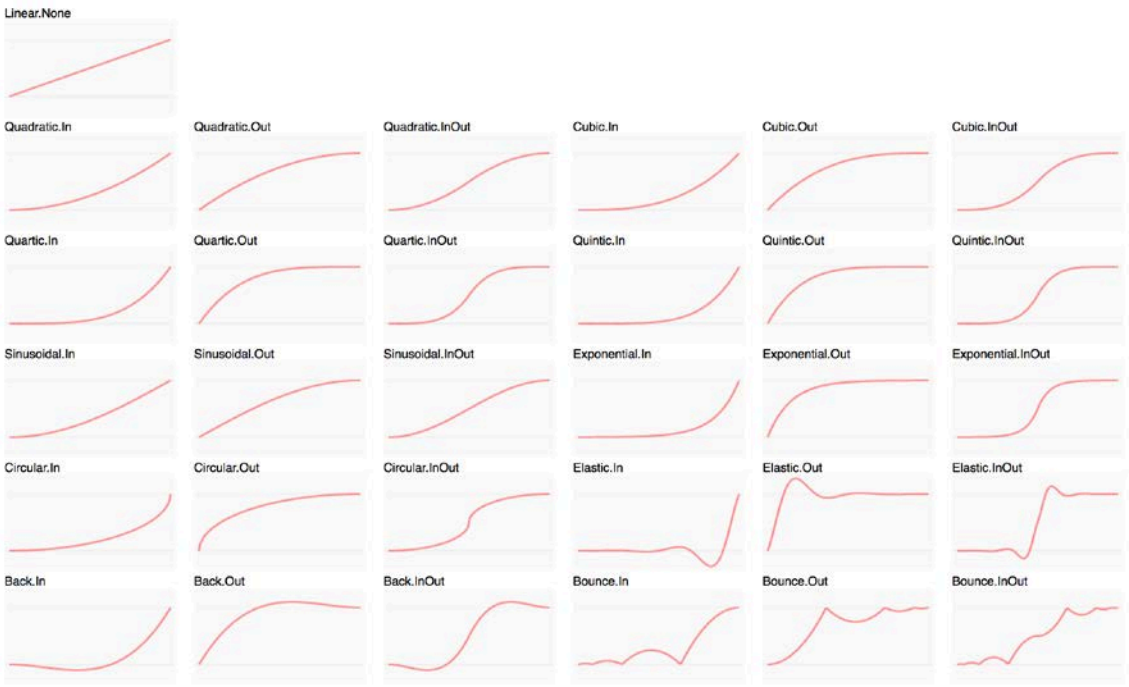
Esimerkkikoodi 5. Kameran liikeradan animointi Tween.js:llä.

Esimerkin cameraEndPosition on objektimuotoinen sijainti halutulle kameran sijainnille. Camera.position muunnetaan objektin arvoihin 600 millisekunnin aikana. Siirroksen jälkeen tweenComplete-muuttuja saa arvon tosi, jota käytetään estämään esimerkiksi kameran muu liikuttelu kesken siirrosten. Animointi ei etene itsestään, vaan Three.js:n animate-silmukassa pitää muistaa kutsua päivitystä jokaisella kuvalla:

```
TWEEN.update();
```

Esimerkkikoodi 6. Tween.js-animaation päivitys jokaisessa kuvassa.

Esimerkissä 5 esiintyvä easing tarkoittaa käyrää, jonka mukaan arvo lasketaan jokaisella ajanhetkellä alku- ja loppupisteiden välillä. Easingia muuttamalla kamera saadaan esimerkiksi kiihtymään hitaasti tai kulkemaan koko matka tasaisella, lineaarisella vauhdilla. Kirjasto sisältää runsaasti eri vaihtoehtoja, jotka on visualisoitu kuvassa 20.



Kuva 20. Tween.js-kirjaston easing-vaihtoehdot.

4.5 Käyttäjätestauksessa tehdyt huomiot

Karttasovelluksen ensimmäiset käyttäjätestaukset tehtiin mobiililaitteilla noin puolessa välissä projektia. Siinä vaiheessa karttaan ei ollut vielä lisätty kuvakkeita eivätkä luokkien nimet näkyneet oikein, joten ne oli kytketty pois päältä. Kommenttia tulikin ensimmäisenä

siitä, että karttaa suurentaessa pitäisi ehdottomasti näkyä luokkien nimet niiden yläpuolella. Lisäksi moni kohteli kartan kontrolleja ”kovakouraisesti”: eräs käyttäjä suurensi karttaa niin nopeasti, että yhden kuvan aikana vauhti ehti kiihtyä niin kovaksi, että kamera meni kartan läpi rakennuksen alapuolelle. Ongelma ratkesi rajoittamalla suurinta mahdollista suurennusnopeutta, jolloin kamera ei pysty yhden kuvan aikana hyppäämään niin suurta matkaa, että se ohittaa ohjelmaan koodatun maksimisuurennuksen.

Käyttäjät myös selkeästi rinnastivat käyttöliittymän muihin karttasovelluksiin kuten Google Mapsiin. Liikkuessaan ympäri karttaa he tekivät nopean lyhyen pyyhkäyksen ja nostivat sormen ylös näytöstä odottaen kartan jatkavan hidastaen vauhtia pyyhkäyksen suuntaan. Tätäkään ei ollut ennen testausta otettu huomioon, ja kamera pysähtyikin kuin seinään, kun sormen nosti näytöstä. Ratkaisuksi tallennettiin viimeinen liikkumisnopeus ennen sormen nostoa muuttujaan ja jatketaan kartan liikutusta tähän suuntaan pienentäen muuttujan arvoa 10 % jokaisessa kuvassa. Näin vauhti hidastuu sulavasti ja kokemus on samankaltainen kuin älypuhelimilla esimerkiksi verkkosivua vierittäessä.

Hiirellä käytettynä ei käytettävyys ollut yhtä intuitiivinen kuin mobiililaitteilla. Vasen hiiren painike pohjassa pitäen kartan liikuttelu onnistui käyttäjiltä ilman ohjeistusta, mutta monikaan ei edes yrittänyt samaa oikealla hiiren painikkeella, joka olisi kääntänyt karttanäkymää. Uskoisin tämän johtuvan siitä, että selaimessa oikea hiiren painike on vakiintunut verkkosivuilla kontekstivalikon avaamiseen.

Muilta osin käyttäjät pitivät sovelluksesta ja osasivat käyttää sitä ilman minkäänlaista ohjeistusta. Mielestäni sovellus toimii silloin hyvin, kun käyttäjät eivät tarvitse neuvoa ohjelman käyttöön. Käyttöliittymä ja painikkeet eivät herättäneet käyttäjien huomiota, ja ne koettiin luonnollisiksi.

Viimeisellä opinnäytetyötä varten tehdyllä versiolla testattiin toimintaa koulun aulassa ohikulkevilla opiskelijoilla. Käyttäjät ohjattiin kuvassa 21 näkyvälle lapulle, ja heitä pyydettiin etsimään sen avulla jokin luokka toiselta puolelta koulua. QR-koodin osoite on muotoa ”http://users.metropolia.fi/~samikol/public/?aula”, jossa lopun ”?aula” on hakuparametri, joka lisää ”olet tässä” -tyylisen ikonin karttaan sekä määrittää sovelluksen automaattisesti avaamaan ensimmäisen kerroksen näkymän, jossa aula sijaitsee.

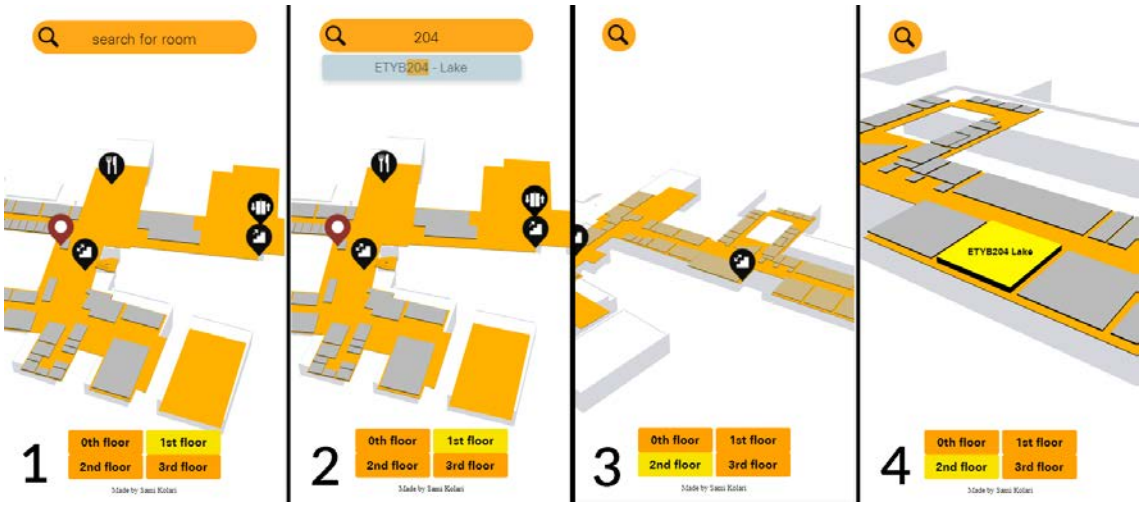
Kuva 21. Koulun aulassa käyttäjätestauksta varten tulostettu QR-koodi.

Ensimmäiseksi haasteeksi osoittautui se, että seitsemästä testaamastani henkilöstä kaikilla kesti useita minuutteja saada kuvan QR-koodi auki. Kaksi henkilöistä muisti joskus ladanneensa QR-koodin lukuun tarvittavan sovelluksen, mutta heillä kesti melko kauan löytää se puhelimestaan. Loput viisi henkilöä joutuivat lataamaan tarvittavan sovelluksen puhelimeensa testiä varten. Eräs henkilöistä kysyi Google Assistantia lukemaan QR-koodin heikoin tuloksin. Applen iPhonen uusimmassa käyttöjärjestelmässä (IOS 11) on oletuskamerasovelluksessa sisäänrakennettu tuki QR-koodien lukemiselle, mutta testaajien Android-pohjaisissa puhelimissa ei tätä ominaisuutta ollut.

Testilaitteilla esiintyi satunnaisia virheitä: Sonyn puhelimessa näytölle ilmestyvä näppäimistö sekoitti kuvakkeiden paikan ja pieninäyttöisissä puhelimissa elementit menivät päällekäin. Joskus, kun hakuun kirjoittaa tekstiä, joka ei tuota yhtään tulosta, katoaa hakupalkki ruudulta täysin. Virheiden aiheuttajat saatiin kuitenkin paikallistettua: esimerkiksi Sonyn puhelimen tapauksessa näyttäisi näytölle ilmestyvä näppäimistö muuttavan selaimen ikkunan pystyresoluutiota. Muissa puhelimissa resoluutio ei muutu, vaikka näppäimistö peittääkin näytön alalaidan. Tähän auttoi WebGL-canvaksen koon uudelleenlaskeminen resoluution vaihtuessa, jolloin WebGL-näkymä vastaa muuttunutta resoluutiota ja HTML-elementit osuvat oikeille kohdille.

Vanhemmilla puhelimilla sovelluksen suorituskyky oli heikko, mutta silti käytettäväksi luokiteltava. Ruudunpäivitysnopeus oli arvioni mukaan alimmillaan noin kymmenen kuvaa sekunnissa. Nämä testaajat eivät kuitenkaan tuntuneet kiinnittävän siihen huomiota, sillä puhelin näytti toimivan kauttaaltaan yhtä hitaasti.

Lisäksi selvisi, että nykyinen reitin esitystapa ei ole riittävä tilanteissa, joissa lähtöpiste tiedetään. Kuvassa 22 on esitetty kuvasarjana nykyinen toimintamalli. Vaihe 1 on lähtötilanne, jossa ”olet tässä” -kuvake on nähtävissä. Testaajilla ei tuntunut olevan vaikeuksia kuvakkeen ymmärtämisessä. Vaiheessa 2 käyttäjä hakee haluttua luokkahuonetta ja näkymä siirtyy toisen kerroksen yleisnäkymään (vaihe 3), josta kamera liikkuu näyttämään halutun huoneen (vaihe 4). Kuten kuvista huomaa, ei viimeisissä vaiheissa ”olet tässä” -ikoni ole enää nähtävissä. Tässä vaiheessa testaajat olivat jo unohtaneet, missä päin kuvake sijaitsi.



Kuva 22. Luokkahuoneen hakemisen vaiheet, kun lähtöpiste tiedetään.

Mielestäni nykyinen toimintatapa on hyvä, kun käyttäjä selaa karttaa tutkimismielessä ja käyttäjän sijaintia ei ole määritelty. Karttaa liikuttelemalla käyttäjä ymmärtää, missä päin koulua valittu huone on. Lisäksi alalaidan korostettu kerroksenvalintapainike tuntui olevan riittävän selvä kertomaan, missä kerroksessa kameran näkymä on.

Reittiohjeita varten lisättiin mahdollisuus piirtää viiva alku- ja loppupisteiden välille. Jos pisteet ovat eri kerroksissa, olisi hyvä näyttää samanaikaisesti kaksi eri kerrosta. Kiinteitä lähtöpisteitä varten voisi karttaan määrittää katselusuunnan, jolloin käyttäjä ja kartta osoittaisivat samaan suuntaan. Tämä selkeyttäisi kartan hahmottamista. Näiden ominaisuuksien toteuttamiseen ei kuitenkaan opinnäytetyön puitteissa rittänyt aikaa.

4.6 Jatkokehitys

Kehityksen edetessä sovellus herätti kiinnostusta Metropolian henkilökunnan keskuudessa. Metropolialla on toiveissa kehittää järjestelmä, jonka kautta näkisi reaaliajassa esimerkiksi luokkahuoneiden käyttötilanteen ja opettajien sijainnit koululla. Lisäksi tarvetta olisi näyttävälle esittelysivulle tulevalle Myllypuron kampukselle. Minut palkattiin jatkokehittämään sovellusta lisäämällä siihen haluttuja toiminnallisuuksia. Projekti on tätä raporttia kirjoittaessani vielä meneillään. Sitä varten olen mallintanut Myllypuron kampuksen piirroskuvien pohjalta (kuva 23) sekä optimoinut Helsingin kaupungin 3D-malleja muiden Metropolian koulurakennusten osalta lisätäkseni myös ne mukaan sovellukseen.

Kuva 23. Myllypuron tuleva kampus visualisoituna. Oikealla näkyvää oranssia ikonia painamalla saa rakennuksesta näkyviin kerroskohtaisen näkymän.

5 Yhteenveto

Insinöörityössä perehdyttiin kauppakeskuksissa käytettäviin opastusnäyttöihin ja niiden teknologioihin. Metropolian Leppävaaran kampukselle tehtiin WEBGL-pohjainen karttasovellus, jossa käyttäjä voi hakea luokkahuoneen sijaintia tai selata karttaa vapaasti. Karttasovelluksen toimivuutta testattiin käyttäjätestauksilla.

Olen tyytyväinen insinöörityön lopputulokseen, ja karttasovelluksesta tuli suurilta osin sellainen, millaiseksi sen suunnittelinkin. Jatkokehitykselle ja hiomiselle löytyy aina tarvetta, ja ominaisuuksista reitin piirto jäikin opinnäytetyön puitteissa tekemättä. Suorituskyky pysyi hyvänä kehityksen loppuun saakka, ja kartta toimii oikein kaikilla testatuilla laitteilla, jopa Samsungin älytelevision sisäänrakennetulla selaimella. Omaa koodia kertyi lopulliseen versioon noin 1 300 riviä, ja koko sovellus vie yhteensä tilaa reilun megatavun verran. 3D-tiedoston tehokkaammalla pakkauksella ja koodin minimoimisella voisi sovelluksen koosta nipistää ehkä vielä kolmanneksen pois, mutta kartta latautuu nykyiselläänkin jo todella nopeasti.

Projektin aikana opin käyttöliittymäsuunnittelusta ja käyttäjätestauksesta. Kosketusnäytön eleiden hyödyntämiseen perehdyin ensimmäistä kertaa, kuten myös CSS-attribuuttien animointiin ja usean piirtotason käyttöön. Jatkokehityksessä olen päässyt syventämään osaamista esimerkiksi jälkiprosessoinnissa FXAA-reunanpehmennyksen ja syväterävyyden simuloinnin osalta. Lisäksi olen kehittänyt laserkeilaskannattujen rakennusten reaaliaikaiseen piirtoon vaaditun optimoinnin osalta melko tehokkaan työnkulun.

Lähteet

1 Craneworks – Referenssejä. Verkkoaineisto. Craneworks. <http://www.craneworks.fi/referenssejae>. Luettu 11.1.2018.

2 Hyper[in] – Transforming reatil real estate. Verkkoaineisto. HyperIn. <http://www.hyperin.com>. Luettu 11.1.2018.

3 Kauppakeskus Kluuvi – Referenssit. Verkkoaineisto. Upto. <http://www.upto.fi/referenssi/kauppakeskus-kluuvi>. Luettu 13.1.2018.

4 Hewitson, Joe. 2013. Three.js and Babylon.js: a Comparison of WebGL Frameworks. Verkkoaineisto. Sitepoint. <https://www.sitepoint.com/three-js-babylon-js-comparison-webgl-frameworks>. Luettu 18.1.2018.

5 Building and running a WebGL project. 2017. Verkkoaineisto. Unity Documentation. <https://docs.unity3d.com/Manual/webgl-building.html>. Luettu 18.1.2018.

6 Devine, Benjamin. WebGL framework comparison. Verkkoaineisto. GitHub. <http://bnjm.github.io/WebGL-framework-comparison/>. Luettu 18.1.2018.

7 Pricing – Clara.io. Verkkoaineisto. Clara.io. <https://clara.io/pricing>. Luettu 19.1.2018.

8 Plans and Pricing. Verkkoaineisto. PlayCanvas. <https://playcanvas.com/plans>. Luettu 19.1.2018.

9 NPAPI deprecation: developer guide. Verkkoaineisto. The Cromium Projects. <https://www.chromium.org/developers/npapi-deprecation>. Luettu 23.1.2018.

10 Krug, Steve. 2014. Don’t make me think! Revisited. USA: New Riders.

11 Kauhanen-Simanainen, Anne. 2003. Informaatioarkkitehtuuri. Helsinki: CIM kustannus.

12 Whitting, Phil. 2017. Do GPS tracking devices work indoors? Verkkoaineisto. <https://www.gps-repeaters.com/blog/do-gps-tracking-devices-work-indoors>. Luettu 30.1.2018.

13 Web Bluetooth. Verkkoaineisto. GitHub. <https://webbluetoothcg.github.io/web-bluetooth/>. Luettu 2.2.2018.

14 Network Information API. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/API/Network_Information_API>. Luettu 3.2.2018.

15 QR Code Essentials. 2011. Verkkoaineisto. Denso. <http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo%3D&tabid=1426&mid=4802>. Luettu 11.2.2018.

16 Graafinen ohjeisto. 2009. Verkkoaineisto. Metropolia. < https://www.metropolia.fi/aineistopankki/graafinen-ohjeisto/> Luettu 15.2.2018.

17 Hildén, Jonatan. 2012. Opastusta Kampissa. Verkkoaineisto. Informaatiomuotoilu. <http://informaatiomuotoilu.fi/2012/03/opastusta-kampissa/>. Luettu 23.2.2018.

18 Saubag digital signage displays & enclosures. 2016. Verkkoaineisto. Saubag. <http://saubag.lv/support/downloads/SAUBAG_2016.pdf> Luettu 24.2.2018.

19 Infrared touch screen technology. Verkkoaineisto. Baanto. <http://baanto.com/infrared-touch-screen-technology-and-ir-phototransistor>. Luettu 24.2.2018.

20 Hamilo, Marko. 2010. Miten kosketusnäyttö toimii? Verkkoaineisto. Tiede. <https://www.tiede.fi/artikkeli/jutut/artikkelit/miten_kosketusnaytto_toimii> Luettu 24.2.2018.

21 Lardinois, Frederic. 2017. Get ready to finally say goodbye to Flash – in 2020. Verkkoaineisto. TechCrunch. <https://techcrunch.com/2017/07/25/get-ready-to-say-goodbye-to-flash-in-2020/> Luettu 25.2.2018.

22 Vincent, Paul. 2017. Flash is done; time for digital signage to fully adopt HTML5. Verkkoaineisto. Sixteen:Nine <http://www.sixteen-nine.net/2017/08/30/flash-is-done-time-for-digital-signage-to-fully-adopt-html5/>. Luettu 25.2.2018.

23 Transparency, Translucency, and Blending. Verkkoaineisto. Khronos Group. <https://www.opengl.org/archives/resources/faq/technical/transparency.htm>. Luettu 3.3.2018.

24 Brown, Alan E. 2016. Resistive vs. Capacitive Screens: Pros & Cons. Verkkoaineisto. Embedded Systems Engineering. <http://eecatalog.com/digital-signage/2016/08/01/resistive-vs-capacitive-screens-pros-cons/>. Luettu 1.4.2018.

25 Khronos Releases Final WebGL 1.0 Specification. 2011. Verkkoaineisto. Khronos Group. <https://www.khronos.org/news/press/khronos-releases-final-webgl-1.0-specification>. Luettu 1.4.2018.

26 Weissflog, Andre. 2016. Thoughts about a WebGL-Next. Verkkoaineisto. The Brain Dump. <https://floooh.github.io/2016/08/13/webgl-next.html>. Luettu 1.4.2018.

27 Kubisch, Christoph. 2016. Transitioning from OpenGL to Vulkan. Verkkoaineisto. Nvidia. <https://developer.nvidia.com/transitioning-opengl-vulkan>. Luettu 1.4.2018.

28 How to install the latest version of DirectX. 2018. Verkkoaineisto. Microsoft. <https://support.microsoft.com/en-us/help/179113/how-to-install-the-latest-version-of-directx>. Luettu 2.4.2018.

29 Tomassetti, Gabriele. 2017. Introduction to WebAssembly: why should we care? Verkkoaineisto. Federico Tomassetti. <https://tomassetti.me/introduction-to-webassembly>. Luettu 2.4.2018.

30 WebAssembly FAQ. Verkkoaineisto. WebAssembly. <http://webassembly.org/docs/faq/>. Luettu 3.4.2018.

31 Rapp, Nick. 2017. Unity and creating content in a Post-Flash World. Verkkoaineisto. Unity Blog. <https://blogs.unity3d.com/2017/07/25/unity-and-creating-content-in-a-post-flash-world/>. Luettu 3.4.2018.

32 chanderprall. Physijs. Verkkoaineisto. GitHub. http://chanderprall.github.io/Physijs/. Luettu 3.4.2018.

33 Perspective and Orthographic Views. Verkkodokumentti. Blender Wiki. <https://wiki.blender.org/index.php/User:Rob221/Doc:2.6/Manual/3D_interaction/Navigating/3D_View>. Luettu 4.4.2018.

34 Kampin liikkeet kartalla. Verkkoaineisto. Kamppi. <https://www.kamppi.fi/fi/node/10960>. Luettu 13.1.2018.